VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Vykresľovanie objemových dát

Volume Rendering

2010 **Bc. Ľubomír Duraj**

Zadanie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa 7.5.2010 .....

Podpis

Na tomto mieste by som rád poďakoval vedúcemu diplomovej práce Ing. Petrovi Gajdošovi, PhD. za odbornú pomoc a konštruktívne pripomienky pri písaní práce.

Abstrakt

Oblasť objemového vykresľovania zaznamenáva v súčasnosti búrlivý posun smerom napred. V dôsledku neustále sa zvyšujúceho výpočtového výkonu dnešných GPU môžu vývojári na celom svete produkovať širokú škálu aplikácií. Táto diplomová práca sa zameriava na zmapovanie možností 3D vizualizácie objemových dát. Sú v nej vysvetlené všetky teoretické a praktické prístupy vedúce k návrhu softwaru produkujúceho snímky dostatočnej kvality. Súčasťou práce je aj samotná aplikácia vykresľujúca objemové dáta.

Kľúčové slová

Objemové vykresľovanie, voxel, dátová mriežka, rekonštrukcia, interpolácia, objemový integrál, Ray Casting, voxelizácia

Abstract

There is a great progress in volume rendering recent few years. Because of better hardware facilities of modern graphics processing units it is easier for IT developers to produce a huge range of applications. The primary goal of this diploma thesis is to provide comprehensive look at 3D visualization possibilities of volume data. It explains all theoretical and practical approaches leading to design and implementation of software which produces images of sufficient quality. This diploma thesis contains also implementation of volume rendering application.

Key words

Volume rendering, voxel, data grid, reconstruction, interpolation, volume rendering integral, Ray Casting, voxelization

Zoznam použitých symbolov a skratiek

BRDF	– Dvojsmerová odrazová distribučná funkcia
CPU	– Centrálna procesorová jednotka
CT	– Počítačová tomografia
CUDA	– Compute Unified Device Architecture
GPU	– Grafická procesorová jednotka
GUI	– Grafické užívateľské rozhranie
MRI	– Magnetická rezonancia
OpenGL	– Open graphics library

Obsah

Prílohy

1. Úvod

V počítačovej grafike sú trojrozmerné objekty najčastejšie vytvárané a reprezentované pomocou povrchovej reprezentácie. Ide najmä o rôzne typy triangulácií (napr. Delaneho triangulácia), kedy sa vytvárajú telesá pozostávajúce s vysokého počtu trojuholníkov pričom vznikajú tzv. trojuholníkové siete (triangle meshes). Mnohokrát je zasa vhodné použiť rôzne typy kriviek a plôch. Obľúbené sú napríklad NURBS plochy, ktoré sa dajú reprezentovať metódou adaptívneho delenia plochy založeného na princípe rekurzívneho delenia plátov (patch splitting). Výstupom tejto metódy sú tzv. NURBS pláty (NURBS patches).

S využitím povrchovej reprezentácie telies ako jedným z modelovacích prístupov sú vizuálne vlastnosti povrchu telies, ako napríklad farba, krivosť alebo schopnosť odrážať svetlo vyjadrené s využitím tieňovania (shading). Ide napríklad o Gouraudovo alebo Phongovo tieňovanie, kedy sú použité rôzne osvetľovacie modely (Phongov osvetľovací model, BRDF a.i.).

Pri nasadení týchto postupov je transport svetla, resp. interakcia svetla s povrchom telies vyhodnotená len v jednotlivých bodoch na ich povrchu. Preto tieto metódy nemajú možnosť opísať transport svetla vo vnútri objektov alebo v iných prostrediach (atmosféra, kvapaliny a pod.). Práve v takýchto prípadoch je vhodné použiť iné modelovacie paradigma – objemové modelovanie (volume modeling, volume rendering).

V porovnaní s povrchovým vykresľovaním, objemové vykresľovanie zahrňuje širokú škálu techník pracujúcich s rôznym typom trojrozmerných skalárnych dát. Prvé algoritmy vykresľujúce objemové dáta siahajú do 80. rokov minulého storočia. Ich najčastejšie uplatnenie možno nájsť najmä v širokej škále vedeckých simulácií a vizualizácií. Objemové dáta (volumetrické dáta) sú získavané z rôznych vedeckých meraní alebo generované numerickými simuláciami. Typickým príkladom sú dáta získané z oblasti medicíny. Ide najmä o snímky rôznych častí ľudského tela z počítačovej tomografie (CT) alebo magnetickej rezonancie (MRI). Ďalšími príkladmi sú geologické dáta, dáta zo seizmických meraní, výsledky simulácií chovania kvapalín, výsledky archeologických meraní a pod. Volumetrické dáta je výhodné vykresľovať aj v oblasti strojárenstva a architektúry – zaťaženie jednotlivých materiálov rôznymi fyzikálnymi veličinami (tlak, teplota, čas). Objemové modelovanie si našlo uplatnenie takisto v hernom priemysle a umení. So zvyšujúcim sa výpočtovým výkonom moderných grafických kariet sa neustále zefektívňujú jednotlivé algoritmy vykresľujúce volumetrické dáta. Aj v tejto oblasti smerujú snahy vývojárov na oblasť využitia shaderov a nasadenia paralelných technológií, ako napr. CUDA.

V jednotlivých kapitolách tejto diplomovej práce sa budem zaoberať práve problematikou objemového vykresľovania. Prácu možno rozdeliť do dvoch hlavných častí. V kapitolách prvej časti podrobne vysvetlím podstatu a princípy, na ktorých funguje objemové vykresľovanie. Táto časť sa mimo iné zaoberá pojmami ako volume rendering integrál, fyzikálny model šírenia svetla v objemových dátach, klasifikácia a pod. Uvediem prehľad základných metód vykresľovania

s prihliadnutím na tie najčastejšie sa vyskytujúce metódy v praxi. V neposlednom rade bude spomenutá tzv. volume rendering pipeline (postupnosť krokov s cieľom vykresľovať objemové dáta).

Druhá časť diplomovej práce bude venovaná návrhu a samotnej implementácii aplikácie v prostredí OpenGL. Podrobne vysvetlím použité postupy a technológie.

2. Teoretický základ a základné prístupy

Táto kapitola sa zameriava mimo iné na fyzikálny model, o ktorý sa opiera objemové vykresľovanie. Bude predstavený jeho podrobný matematický popis spolu s jeho prípustnými aproximáciami, ktoré sa osvedčili v praxi pri tzv. real – time objemovom vykresľovaní. Ďalšie podkapitoly sa budú podrobne venovať charakteristike objemových dát, rôznym typom dátových mriežok, na ktorých je možné reprezentovať diskrétne objemové dáta a problému rekonštrukcie diskrétnych dát na spojité. Budú vysvetlené pojmy ako voxel, volume rendering pipeline a predstavené základné techniky objemového vykresľovania.

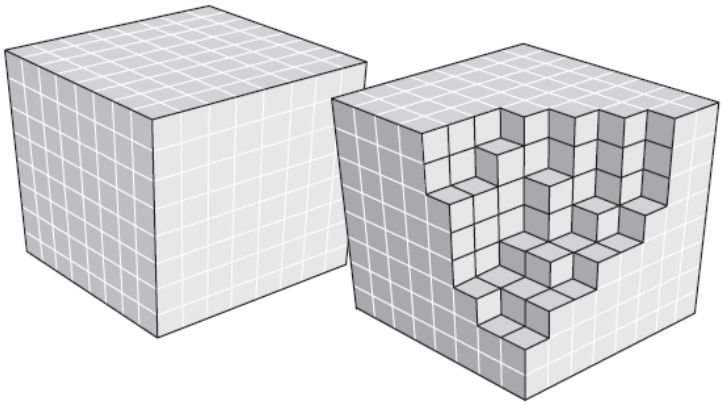
2.1. Fyzikálny model transportu svetla

Hlavnou myšlienkou tohto modelu je snaha modelovať transport svetla v opticky aktívnom prostredí (napr. plynné prostredie). Preto má objemová grafika podobne ako „štandardná“ počítačová grafika za cieľ simulovať propagáciu svetla v nejakom participujúcom médiu s cieľom produkovať snímky dostatočnej kvality za použitia určitého osvetľovacieho modelu. Z toho vyplýva, že interakcia svetla a média musí byť vyhodnocovaná v každom bode trojrozmerného objemu vyplneného daným participujúcim médiom. Energia svetla môže byť v takomto prostredí absorbovaná, svetlo môže byť rozptýlené alebo vyžarované samotným médiom. Presne vyhodnotiť vzájomné interakcie svetla a prostredia v každom jeho bode je výpočtovo náročná úloha. Existuje však viacero techník a aproximácií, ktoré tento problém riešia.

Priame zobrazovanie objemu (angl. direct volume rendering) ako jedna zo základných metód vykresľovania má za cieľ vizuálne extrahovať informáciu z trojrozmerného skalárneho priestoru, resp. trojrozmerných skalárnych dát. Túto činnosť možno reprezentovať ako funkciu:

$$\phi : R^3 \rightarrow R$$

Ide teda o zobrazenie z trojrozmerného priestoru na skalárnu (číselnú) hodnotu. Spomenuté trojrozmerné skalárne dáta majú pôvod v rôznych meraniach a vedeckých simuláciách a sú definované na rôznych typoch diskrétnych mriežok v podobe vzorkou (angl. samples). Obrázok 1 ilustruje takéto objemové dáta reprezentované na pravidelnej pravouhlej diskrétnej mriežke. Takáto diskretizácia však vedie k problému rekonštrukcie funkcie ϕ vo všetkých bodoch trojrozmerného priestoru. Funkcia ϕ je totiž spojitá, je teda definovaná v ľubovoľnom bode trojrozmernej domény a nie iba v jednotlivých diskrétnych vzorkách. Problémom rekonštrukcie funkcie ϕ sa budem zaoberať v kapitole 2.3. Na tomto mieste je splnený predpoklad existencie danej rekonštrukcie a teda funkcia ϕ je definovaná na kompletnej doméne. Priame zobrazovanie objemu mapuje trojrozmerný skalárny priestor na fyzikálne veličiny (tlak, teplota, hustota a pod.), ktoré popisujú interakciu svetla v danom bode trojrozmerného



Obrázok č. 1: Objemové dáta reprezentované pravidelnou diskrétnou mriežkou

priestoru. Máme teda k dispozícii popis priestorového rozloženia hodnôt ľubovoľnej fyzikálnej veličiny v participujúcom médiu. Daný popis následne slúži na výpočet osvetlenia v konkrétnom bode, presnejšie na vyhodnotenie interakcie svetla v danom bode priestoru. Celý tento proces sa nazýva klasifikácia a je založený na koncepte tzv. transfer functions. Klasifikácia je detailne vysvetlená v kapitole 3.1.

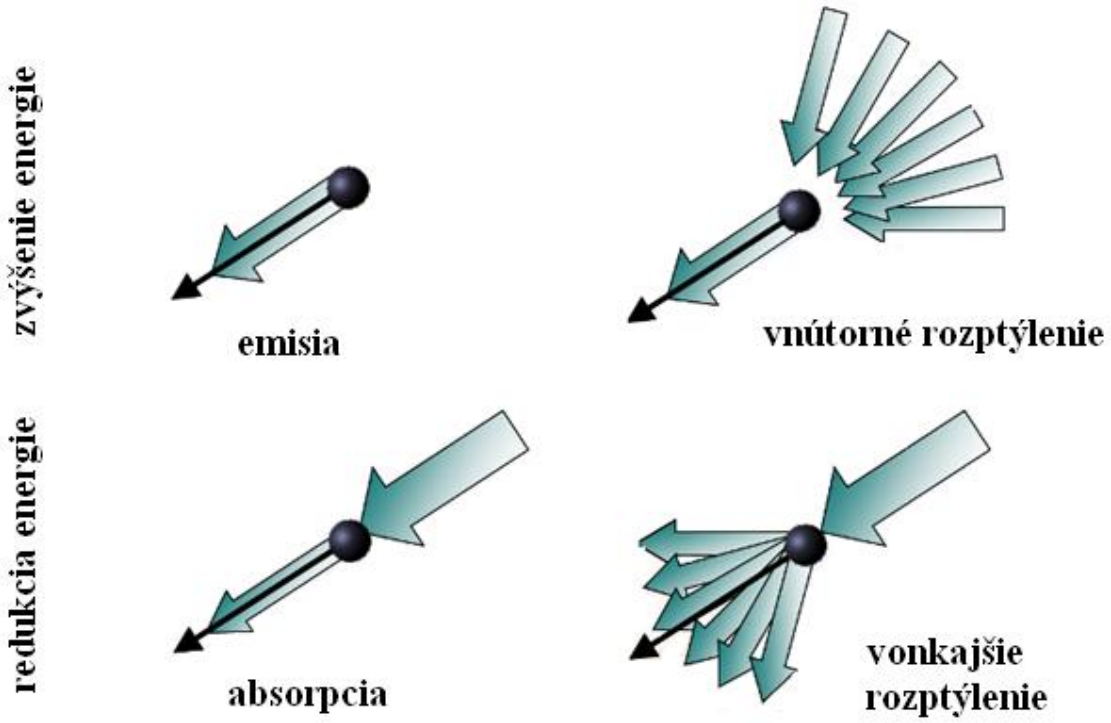
2.1.1. Rovnica transportu svetla v priestore

V tradičnej počítačovej grafike sme predpokladali, že šírenie svetla, resp. jeho trajektória nie je ničím rušená alebo ovplyvňovaná. Priestor bol dokonale prázdny a fotón vyžiarený zo svetelného zdroja, odrazený od povrchu objektu, či prejdený priehľadným objektom, putoval po priamke. S predchádzajúcej časti je však jasné, že v prípade objemového vykresľovania nie je tento predpoklad úplne splnený. Svetlo sa šíri v opticky aktívnom prostredí. Nejaké médium sa zúčastňuje jeho transportu a „bráni“ mu v ňom. Preto je dôležité sledovať a matematicky korektne popísať interakciu svetla a média v priestore. Všeobecne môže dôjsť k nasledujúcim typom interakcií:

- **emisia** – samotné médium aktívne vyžaruje svetlo, čím zvyšuje energiu žiarenia, v prírode ide napr. o horúce plyny vyžarujúce svetlo premieňaním tepelnej energie na energiu žiarenia
- **absorpcia** – materiál môže absorbovať svetlo premenou svetelnej energie na teplo, v takomto prípade je energia svetla redukovaná
- **rozptýlenie** – svetlo môže byť rozptýlené médiom, kedy dochádza ku zmene smeru jeho šírenia v priestore, v prípade, kedy vlnová dĺžka svetla nie je rozptýlením zmenená, tento proces nazývame elastické rozptýlenie, opak sa nazýva neelastické rozptýlenie, v tomto texte sa však vždy bude jednať o elastické rozptýlenie

Absorpcia, emisia a rozptýlenie svetla ovplyvňujú množstvo svetelnej energie pozdĺž trajektórie šírenia svetla, resp. svetelného paprsku. Na obrázku č. 2 je možné vidieť všetky tri typy interakcií

svetla v priestore spolu s prípadmi, kedy je svetelná energia redukovaná alebo naopak dôjde k jej zvýšeniu.



Obrázok č. 2: Základné typy interakcií svetla v médiu

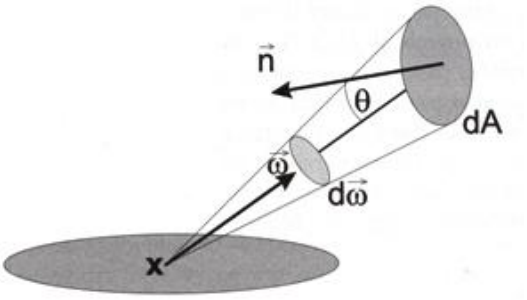
Svetelnú energiu môžeme charakterizovať vo forme tzv. žiarivosti I, ktorá je definovaná ako energia žiarenia Q na jednotkovú plochu A, do priestorového uhla Ω, za jednotku času t:

$$I = \frac{dQ}{dA_p \, d\Omega \, dt} \tag{2.1}$$

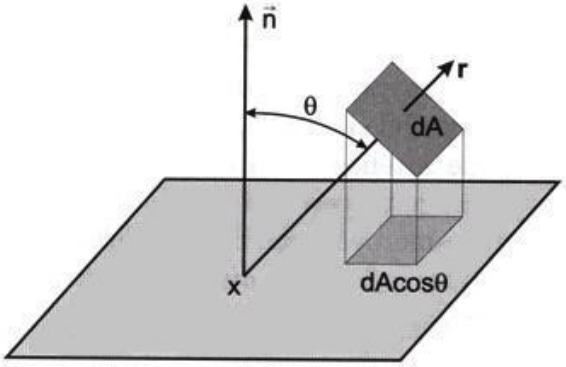
Presnejšie povedané, žiarivosť I udáva prijímaný alebo vyžiarený výkon na jednotkovom priestorovom uhle Ω na jednotku kolmo premietnutej plochy dA_p. Žiarivosť I je možné si predstaviť ako veličinu udávajúcu počet fotónov prichádzajúcich alebo vyžiarených v určitom smere za jednotku času t prechádzajúcu priemetom diferenciálnej plochy dA, ktorá je kolmá na tento smer. Index p v predchádzajúcej rovnici teda indikuje, že ide o veľkosť plochy, ktorá je priemetom diferenciálnej plochy dA orientovanej kolmo na smer svetelného paprsku. Preto platí nasledovné:

$$dA_p = dA \cdot \cos\theta, \tag{2.2}$$

kde θ je uhol medzi smerom paprsku dopadajúceho kolmo na diferenciálnu plochu dA a normálou premietnutej plochy dA_p. Žiarivosť I sa niekedy označuje aj ako intenzita alebo radiancia. Obyčajne označuje farbu paprsku. Pre lepšiu názornosť:



Obrázok č. 3: Výpočet radiancie v bode x pri smere ω svetelného paprsku



Obrázok č. 4: Výpočet radiancie s naznačením priemetu diferenciálnej plochy dA

Prítomnosť participujúceho média ovplyvňuje žiarivosť I pozdĺž svetelného paprsku. Absorpcia spôsobuje redukciu svetelnej energie, kým emisia zvyšuje energiu žiarenia. Rozptýlenie má dvojaký efekt. V prípade tzv. vnútorného rozptýlenia je dodatočná energia presmerovaná do smeru svetelného paprsku čím dôjde ku zvýšeniu svetelnej energie. Môže však nastať prípad, ktorý sa označuje ako vonkajšie rozptýlenie. Energia svetelného parpsku je redukovaná rozptýlením svetla do rozličných smerov v priestore. Všetky tieto javy môžeme sledovať na obrázku č. 2.

Kombináciou absorpcie, emisie a rozptylu je možné získať nasledujúcu rovnicu šírenia svetla v priestore:

$$\boldsymbol{\omega \cdot \nabla_x I(x,\omega) = -\chi I(x,\omega) + \eta} \tag{2.3}$$

Výraz $\omega \cdot \nabla_x I$ je skalárny súčin medzi smerom svetla ω a gradientom žiarivosti I s ohľadom na pozíciu x v priestore. Nabla operátor $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$ je skráteným zápisom operátoru označujúceho gradient nejakej fyzikálnej veličiny (v tomto prípade žiarivosti svetla). Tento skalárny súčin vyjadruje smerovú deriváciu a to v smere šírenia svetla. V prípade parametrizácie svetelného paprsku funkciou arc z jeho dĺžky s, môžeme nahradiť výraz $\omega \cdot \nabla_x I$ deriváciou dI/ds. Člen χ nazývame úplný absorpčný koeficient a vyjadruje tú pomernú časť svetelnej energie, ktorá je pohltená participujúcim médiom. Člen η sa nazýva úplná emisia a charakterizuje do akej miery je svetelná energia zvýšená pri prechode svetelného paprsku priestorom.

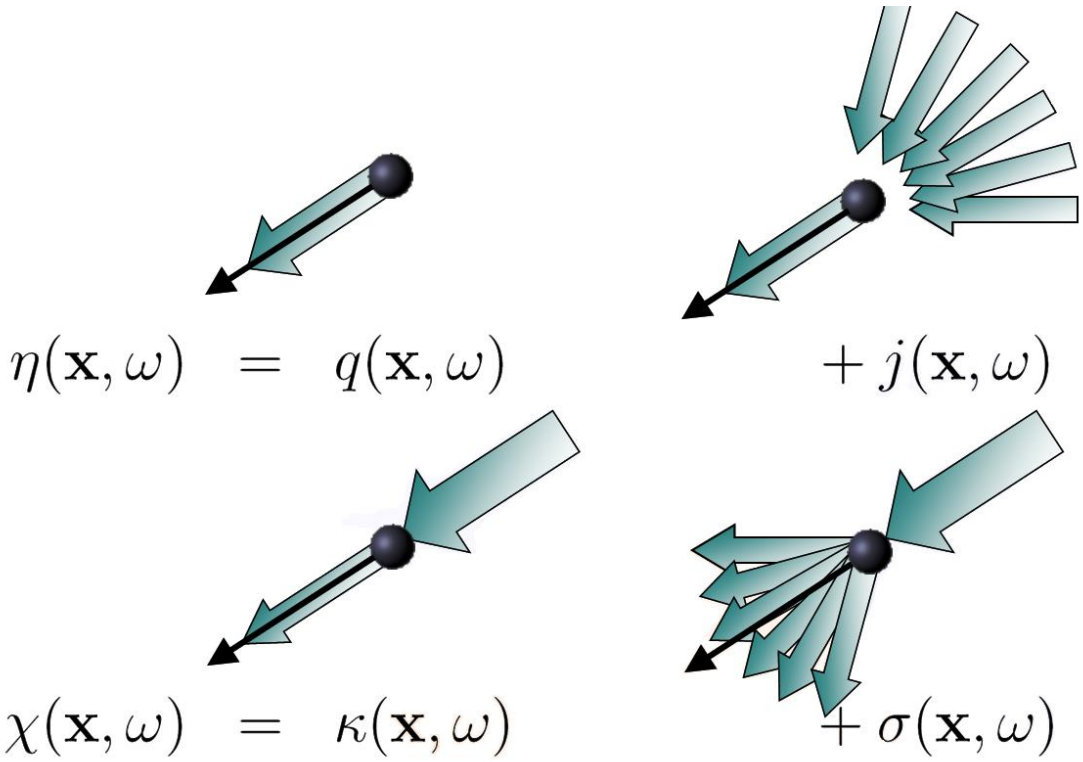
Úplný absorpčný koeficient χ pozostáva z dvoch častí: skutočného absorpčného koeficientu κ (reprezentujúci konverziu svetelnej energie na teplo) a koeficientu rozptýlenia σ, ktorý reprezentuje energetické straty spôsobené vonkajším rozptýlením. Preto úplný absorpčný koeficient môže byť prepísaný do nasledujúcej podoby:

$$\chi = \kappa + \sigma \tag{2.4}$$

Analogicky, úplná emisia η môže byť rozdelená na zdrojový člen q, ktorý reprezentuje čistú emisiu (tepelná energia sa mení na svetelnú energiu) a koeficient vnútorného rozptýlenia j charakterizujúci dodatočný svetelný energetický zisk z vnútorného rozptýlenia. Platí teda nasledujúci vzťah:

$$\eta = q + j \tag{2.5}$$

Na obrázku č. 5 je názorný popis spomenutých členov a k nim priradený typ interakcie svetla v médiu.



Obrázok č. 5: Interakcia svetla v priestore s príslušnými koeficientmi

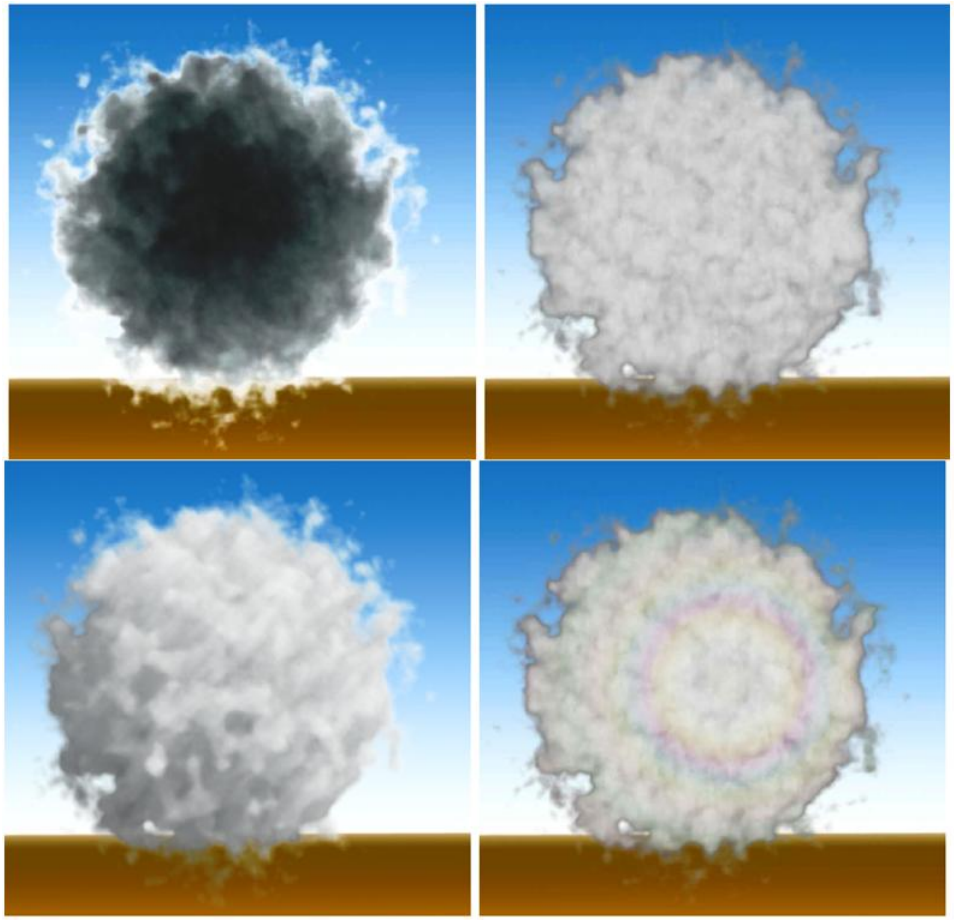
Všetky spomenuté veličiny χ, η, κ, σ, q, j v rovnici 2.3 závisia na pozícii x v priestore a smere svetelného paprsku ω. V prípade členov κ, σ a q ide o optické materiálové vlastnosti, ktoré sú priamo priradené v procese klasifikácie.

Veličina j predstavuje dodatočný energetický zisk z externých svetelných zdrojov prichádzajúci z rôznych smerov do pozície x v uvažovanom priestore. Vzťah 2.6 definuje množstvo tejto energie:

$$j = \frac{1}{4\pi} \int_{guľa} \sigma(x,\omega')p(x,\omega',\omega)I(x,\omega')d\Omega' \tag{2.6}$$

Energetický príspevok z konkrétneho externého zdroja dopadajúci do uvažovaného bodu x v smere ω' reprezentuje člen $I(x,\omega')$. Vzhľadom k faktu, že takýchto zdrojov je v scéne viac, je potrebné akumulovať ich príspevky integráciou vo všetkých smeroch ω'. Integrácia prebieha cez jednotkovú guľu, pretože nás zaujíma celkové množstvo energie dopadajúce z priestoru (zo všetkých smerov ω') cez guľovú plochu do konkrétneho bodu x. Člen $I(x,\omega')$ je pri integrácii prenásobený koeficientom σ a fázovou funkciou p, ktorá predstavuje pravdepodobnosť, že svetelná energia, resp. paprsok prichádzajúci zo smeru ω' je rozptýlený, resp. dôjde k zmene jeho trajektórie do nového smeru ω. Rola fázových funkcií pri popise transportu svetla v participujúcom médiu je podobná tej, ktorú zohráva BRDF funkcia v klasickom povrchovom (angl. surface) vykresľovaní. BRDF totiž poskytuje formálny aparát, ktorý umožňuje opísať schopnosti materiálov odrážať alebo absorbovať svetlo. Fázové funkcie v objemovom vykresľovaní takisto popisujú následnú distribúciu svetla v okamihu,

kedy dôjde k jeho rozptýleniu. Preto sú tieto funkcie dôležitými optickými vlastnosťami participujúceho média. Odlišné materiály vyskytujúce sa v objeme môžu mať odlišné fázové funkcie, čo vedie k odlišnému vzhľadu výsledných snímok – tak ako odlišné materiálové vlastnosti popísané BRDF funkciami vedú k rôznemu výsledku v prípade povrchového vykresľovania. Na nasledujúcom obrázku možno sledovať Henyey – Greensteinovu fázovú funkciu použitú pri vykresľovaní oblakov.



Obrázok č. 6: Použitie fázovej funkcie pri vykresľovaní oblakov

Na ľavom hornom obrázku je oblak osvetlený zozadu zatiaľ čo na pravom hornom obrázku spredu. Okraje oblakov sú v prvom prípade presvetlené, v druhom prípade sú tmavé. Je to spôsobené tým, že Henyey – Greensteinova funkcia v oblasti okrajov rýchlo nadobúda lokálne extrémy. Obrázok vpravo dole je osvetlený pod 45 stupňovým uhlom vzhľadom k pozorovateľovi. Pravý dolný obrázok bol vygenerovaný s použitím Henyey – Greensteinovej fázovej funkcie v kombinácii s Mie fázovou funkciou, ktorá spôsobila v obrázku tzv. glory efekt. V ďalších statiach sa predpokladá, že použitá fázová funkcia je normalizovaná, t.j. platí:

$$\frac{1}{4\pi}\int_{guľa} p(x,\omega',\omega)d\Omega' = 1.$$

Koeficient $\frac{1}{4\pi}$ v predchádzajúcom vzťahu a vzťahu 2.6 ruší vplyv činiteľa 4π, ktorý sa vyskytuje pri integrácii cez jednotkovú guľu.

Vychádzajúc zo vzťahu 2.3 a pri dosadení odvodených koeficientov pre emisiu, absorpciu a rozptyl (vnútorný i vonkajší) je možné získať **kompletnú rovnicu popisujúcu šírenie svetla v priestore:**

$$\boldsymbol{\omega} \cdot \boldsymbol{\nabla}_x \boldsymbol{I}(\boldsymbol{x}, \boldsymbol{\omega}) = -\big(\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{\omega}) + \boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{\omega})\big)\boldsymbol{I}(\boldsymbol{x}, \boldsymbol{\omega}) + \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{guľa} \boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{\omega}')\, \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{\omega}', \boldsymbol{\omega}) \boldsymbol{I}(\boldsymbol{x}, \boldsymbol{\omega}') d\boldsymbol{\Omega}' \tag{2.7}$$

Rovnica 2.7 je dlhšou verziou rovnice 2.3. Táto práca sa však zameriava na efektívne metódy vykresľovania a určenia radiancie *I* z predchádzajúcej rovnice. Preto sa veľmi často pristupuje k prijateľným aproximáciám, kedy sa nevyužíva kompletná rovnica 2.7, ale len jej podmnožina. Týmto prístupom sa dosiahne nižšia výpočtová zložitosť. Možnosti, ktoré sú k dispozícii pri výbere prípustnej aproximácie, resp. pri nasadení zredukovaného modelu sú náplňou nasledujúcej podkapitoly.

2.1.2. Optické modely ako prístup k vhodnej aproximácii

Vzhľadom k tomu, že riešenie kompletnej rovnice šírenia svetla v priestore je výpočtovo náročná úloha, využívajú sa často zjednodušené a tým pádom efektívnejšie modely. Ich hlavným cieľom je zredukovať počet členov v rovnici 2.7 alebo ich vhodne zjednodušiť. Často sa používajú nasledujúce modely:

- **absorpčný model** – objem pozostáva z chladného, dokonale čierneho materiálu, ktorý absorbuje svetlo, nedochádza k emisii ani k žiadnej forme rozptylu
- **emisný model** – objem pozostáva z „plynu“, ktorý len vyžaruje svetlo a je kompletne priehľadný, nedochádza k absorpcii ani rozptylu
- **emisno-absorpčný model –** ide o najčastejšie sa vyskytujúci model v objemovom vykresľovaní, kedy je svetlo z externých zdrojov objemom absorbované a objem aj vyžaruje svetelnú energiu, nedochádza však k nijakej forme rozptylu
- **+ ďalšie komplexnejšie modely**

Ako už bolo spomenuté, emisno-absorpčný model je najčastejšie sa vyskytujúci aproximačný model v oblasti objemového modelovania, resp. vykresľovania. Poskytuje kvalitné výsledky a efektívnu výpočtovú zložitosť. Rovnica tohto modelu ma nasledujúcu podobu:

$$\boldsymbol{\omega} \cdot \boldsymbol{\nabla}_x \boldsymbol{I}(\boldsymbol{x}, \boldsymbol{\omega}) = -\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{\omega})\boldsymbol{I}(\boldsymbol{x}, \boldsymbol{\omega}) + \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\omega}) \tag{2.8}$$

Označuje sa ako **rovnica objemového vykresľovania** (angl. volume rendering equation). Vznikla z rovnice 2.7 vypustením členov reprezentujúcich rozptyl, ktorý sa v emisno-absorpčnom modeli nepripúšťa. Ide o rovnicu v diferenciálnej podobe – špecifikuje transport svetla prostredníctvom zmien v žiarivosti (žiarivosť sa mení pozdĺž svetelného paprsku). V prípade, že uvažujeme len jeden svetelný paprsok, rovnica 2.8 má tvar:

$$\frac{dI(s)}{ds} = -\kappa(s)I(s) + q(s), \tag{2.9}$$

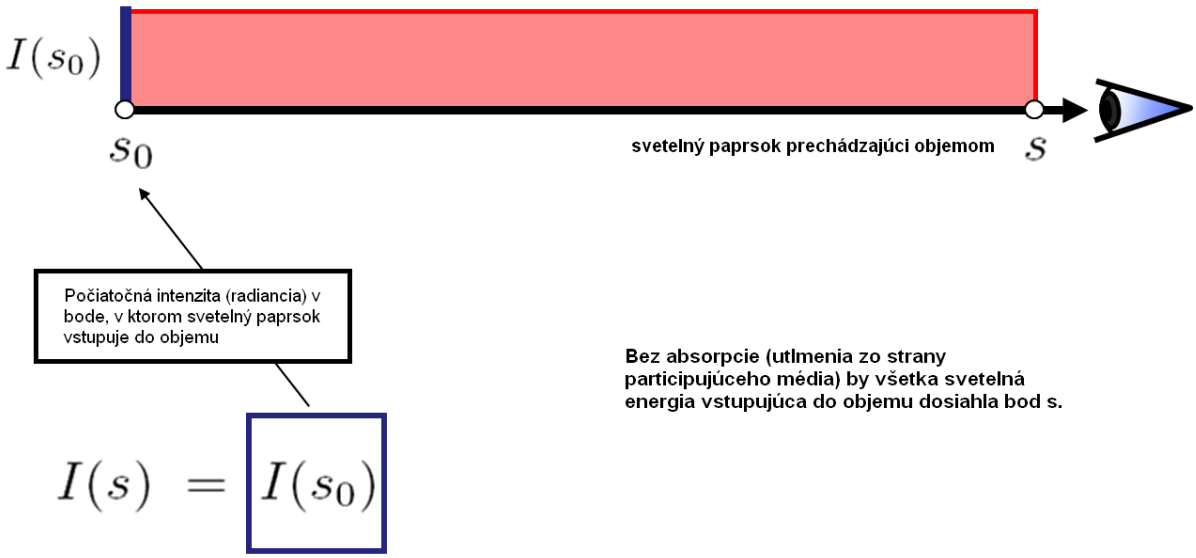
pričom pozícia je vyjadrená prostredníctvom parametra s.

2.1.3. Integrál objemového vykresľovania

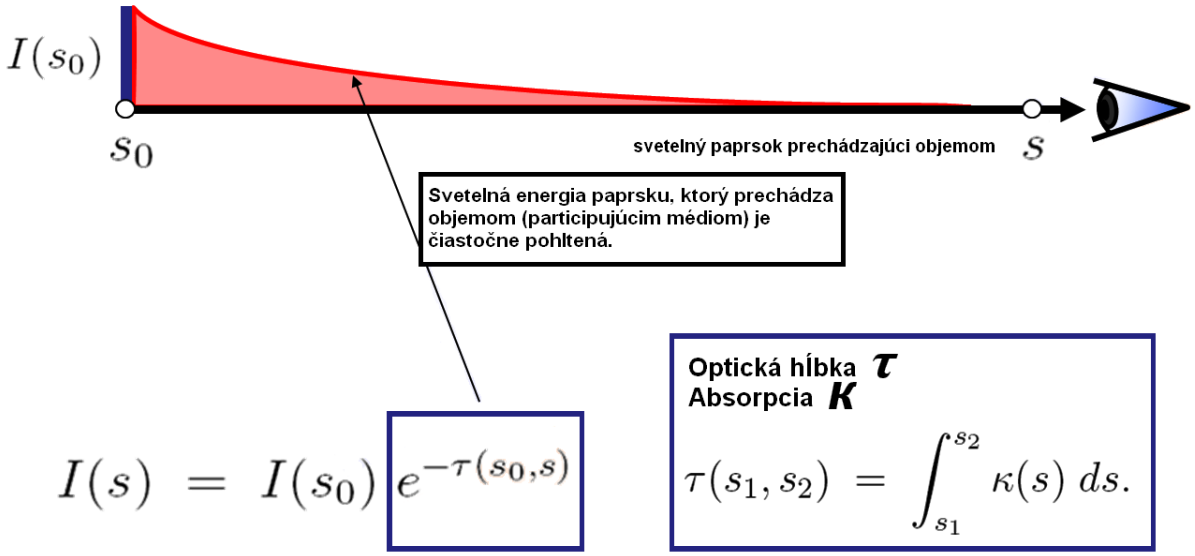
Odvodenie integrálu pre zobrazovanie objemov vychádza z rovnice objemového vykresľovania v jej diferenciálnej podobe – rovnica 2.9. Predpokladá sa, že daným objemom, resp. participujúcim médiom prechádza svetelný paprsok. Ten vstupuje do objemu v bode s_0 a opúšťa ho v bode s. Integrál objemového vykresľovania určuje hodnotu radiancie, resp. intenzity svetla práve v bode s. Táto hodnota – označuje sa $I(s)$, následne slúži v ďalších krokoch tzv. volume rendering pipeline a to najmä v procese klasifikácie. Skúmaný integrál má nasledujúcu podobu:

$$I(s) = I(s_0)e^{-\int_{s_0}^{s}\kappa(t)dt} + \int_{s_0}^{s} q(\tilde{s})e^{-\int_{\tilde{s}}^{s}\kappa(t)dt}\,d\tilde{s} \tag{2.10}$$

Prvá časť integrálu vyjadruje stav, kedy dochádza k čiastočnému utlmeniu, resp. absorpcii svetelnej energie paprsku v dôsledku jeho prechodu participujúcim médiom. Túto vlastnosť názorne predstavujú nasledujúce obrázky:



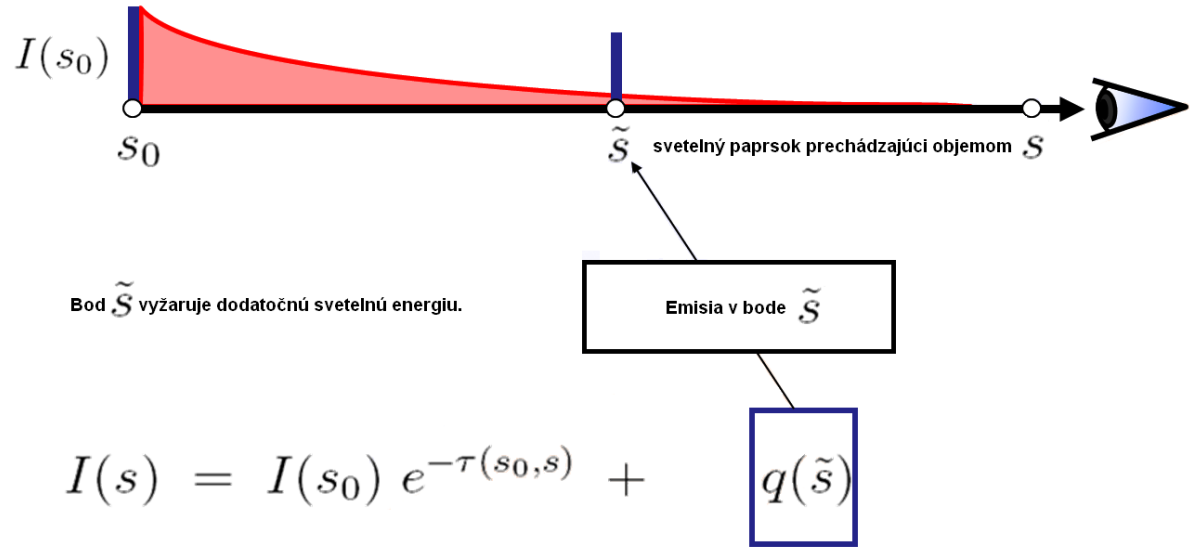
Obrázok č. 7: Svetelná energia paprsku bez jej absorpcie objemom



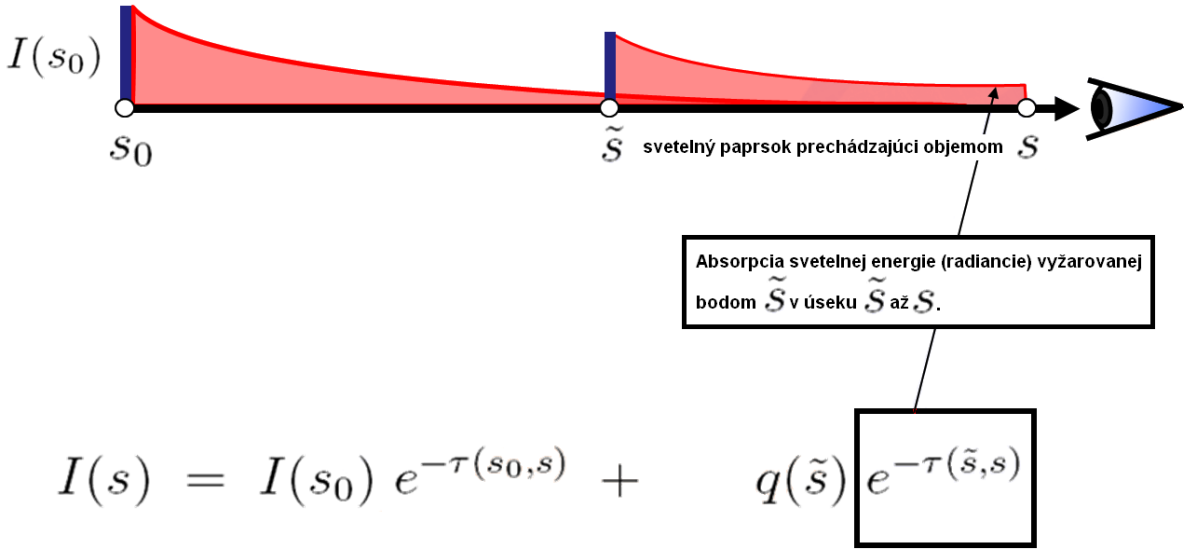
Obrázok č. 8: Svetelná energia paprsku čiastočne pohltená objemom

Výraz $\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(t)dt$ sa označuje ako optická hĺbka medzi pozíciami s_1 a s_2 v objeme. Vyjadruje ako dlho sa môže svetlo šíriť konkrétnym participujúcim médiom do okamihu, kým nie je úplne absorbované. Malé hodnoty tejto veličiny znamenajú, že médium je viac menej priehľadné, zatiaľ čo vysoké hodnoty predstavujú matné, resp. nepriehľadné prostredie.

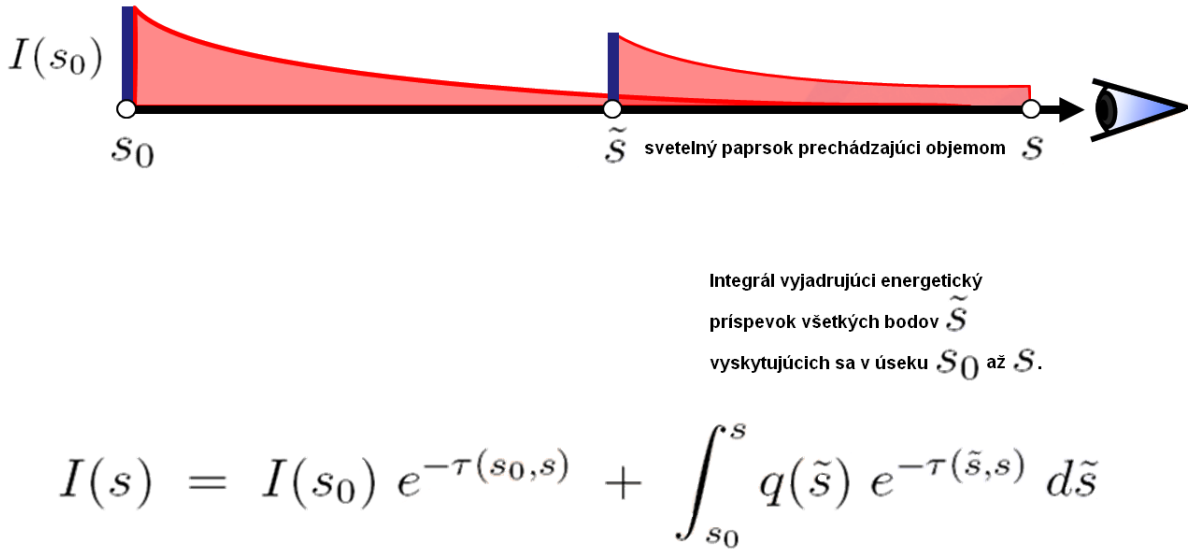
Vzhľadom k faktu, že popisovaný objemový integrál sa opiera o emisno – absorpčný model, je nutné zohľadniť energetické príspevky pozdĺž dráhy šírenia svetelného paprsku, t.j. samotné médium vyžaruje v jednotlivých bodoch dodatočnú svetelnú energiu. Tento jav reprezentuje druhá časť objemového integrálu. Na obrázkoch č. 9, 10 a 11 je možné sledovať takéto správanie.



Obrázok č. 9: V bodoch objemu dochádza k emisii svetelnej energie



Obrázok č. 10: Čiastočné pohltenie energie vyžiarenej z konkrétneho bodu média



Obrázok č. 11: V médiu sa vyskytuje viacero bodov, ktoré vyžarujú dodatočnú energiu

V predchádzajúcich častiach textu bola spomenutá tzv. optická hĺbka a jej vzťah k priehľadnosti, resp. nepriehľadnosti prostredia, ktorého sa táto veličina týka. Priehľadnosť média je teda možné definovať nasledujúcim spôsobom:

$$T(s_1, s_2) = e^{-\tau(s_1,s_2)} = e^{-\int_{s_1}^{s_2} \kappa(t)dt}$$

Po definícii priehľadnosti participujúceho média je možné získať mierne odlišný zápis pre integrál objemového vykresľovania. Jeho podoba je nasledovná:

$$I(s) = I(s_0)T(s_0, s) + \int_{s_0}^{s} q(\tilde{s})T(\tilde{s}, s)d\tilde{s} \tag{2.11}$$

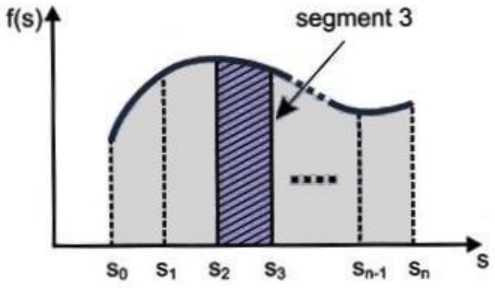
Tento tvar rovnice objemového integrálu bude často využívaný v ďalších častiach tejto práce.

2.2. Vyčíslenie objemového integrálu - diskretizácia

S predchádzajúcich častí textu je jasné, že hlavnou úlohou objemovej grafiky je snaha efektívne a relatívne presne vypočítať v jednotlivých aplikáciách objemový integrál, rovnica 2.10. Daný integrál nemôže byť v praxi vyčíslený analyticky. Namiesto toho sa bežne používajú numerické metódy slúžiace ako vhodná aproximácia k danému integrálu.

2.2.1. Rozdelenie domény objemového integrálu na intervaly

Hlavnou myšlienkou je rozdeliť integračnú doménu na n susedných intervalov. Vzniknuté intervaly sú jednoznačne určené svojou pozíciou $s_0 < s_1 \ldots < s_{n-1} < s_n$, pričom platí, že s_0 je počiatočným bodom domény integrálu a $s_n = D$ je jej koncový bod. Na nasledujúcom obrázku je možné pozorovať rozdelenie domény na jednotlivé intervaly.



Obrázok č. 12: Rozdelenie integračnej na jednotlivé segmenty

V prípade zamerania sa na transport svetla v intervale $[s_{i-1}, s_i]$, pre radianciu, resp. intenzitu v bode s_i platí nasledujúci vzťah:

$$I(s_i) = I(s_{i-1})T(s_{i-1}, s_i) + \int_{s_{i-1}}^{s_i} q(s)T(s, s_i)ds \tag{2.12}$$

Pre lepšiu prehľadnosť v ďalšom texte je vhodné zaviesť nové označenie pre priehľadnosť participujúceho média v danom intervale a radianciu, resp. intenzitu[1] daného i-teho intervalu:

$$T_i = T(s_{i-1}, s_i), \qquad c_i = \int_{s_{i-1}}^{s_i} q(s)T(s, s_i)ds \tag{2.13}$$

[1] Pod pojmom intenzita sa v ďalšom texte rozumie farebný príspevok c konkrétneho intervalu v podobe RGB

Vyšrafovaný interval na obrázku č. 12 ilustruje výsledok integrácie nad jedným intervalom ako obsah vyšrafovanej plochy. V bode, kde svetelný paprsok opúšťa objem, teda v koncovom bode $s_n = D$ integračnej domény, platí pre radianciu, resp. intenzitu nasledujúce:

$$I(D) = I(s_n) = I(s_{n-1})T_n + c_n = (I(s_{n-2})T_{n-1} + c_{n-1})T_n + c_n = \cdots, \tag{2.14}$$

kde $0 < i \leq n$. Vzťah 2.14 je rovnica v rekurentnom tvare. Postupnosťou jednoduchých úprav a substitúcií je možné previesť túto rovnicu do podoby:

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j, \tag{2.15}$$

pričom platí, že $c_0 = I(s_0)$.

Problém samotného výpočtu objemového integrálu však ešte stále nie je doriešený. Integračná doména bola síce rozdelená na n diskrétnych intervalov a teda rovnica 2.15 už môže byť efektívne vypočítaná prostredníctvom CPU alebo GPU. Stále však nie je doriešený problém výpočtu priehľadnosti T_i a intenzity c_i konkrétneho intervalu. K ich výpočtu sa bežne používajú rôzne aproximačné postupy, pričom najčastejšie sa k aproximácii objemového integrálu používa technika nazvaná odhad Riemannovým súčtom. Ide o klasickú definíciu určitého integrálu, kedy je funkcia určená na integráciu aproximovaná, resp. po častiach v jednotlivých intervaloch dĺžky $\Delta x = (D - s_0)/n$ nahradená konštantnou funkciou. Integrál nad jedným konkrétnym intervalom $[s_{i-1}, s_i]$ korešponduje k obsahu obdĺžnikovej plochy definovanej funkčnou hodnotou konštantnej funkcie v bode s_i a dĺžkou Δx daného intervalu, viď. obrázok č. 13. Integrál funkcie v celej integračnej doméne sa potom rovná súčtu obsahov všetkých takto vzniknutých obdĺžnikových plôch nad jednotlivými ekvidištantnými úsekmi. Pri použití takejto aproximačnej techniky platí pre priehľadnosť T_i participujúceho média v konkrétnom intervale i nasledovné:

$$T_i \approx e^{-\kappa(s_i)\Delta x} \tag{2.16}$$

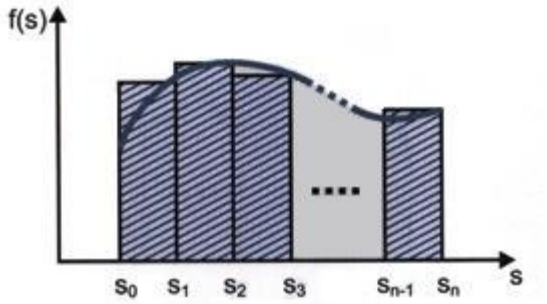
a pre intenzitu v tom istom intervale platí:

$$c_i \approx q(s_i)\Delta x. \tag{2.17}$$

Práve popísaný prístup k výpočtu objemového integrálu je najčastejšie sa vyskytujúca aproximácia v konkrétnych aplikáciách. Typickým príkladom je tzv. texture – based volume rendering. Tento algoritmus bude podrobne predstavený v kapitole venovanej jednotlivým technikám objemového vykresľovania. Existujú však aj iné spôsoby ako vyhodnotiť objemový integrál

v participujúcom médiu. Ide napr. o metódu Monte Carlo. Jej výhodou je, že efektívne potlačuje anti – aliasing spôsobený pravidelným vzorkovaním zobrazovaných skalárnych dát.

Presnosť a rýchlosť výpočtu objemového integrálu je možné v praxi spresniť a zrýchliť. K tomuto účelu sa používa technika nazvaná pre – integrácia. Ide v podstate o to, že sa všetky možné výsledky objemového integrálu pre každý interval vopred s vysokou presnosťou vypočítajú podľa vzťahov 2.13. Počas behu aplikácie dôjde už len k výpočtu podľa vzťahu 2.15, kedy sa do tejto rovnice dosadia predpočítané hodnoty z pre – integrácie.



Obrázok č. 13: Aproximácia integrálu Riemannovým súčtom

2.2.2. Kompozícia

Kompozícia je základom pre iteratívne vyčíslenie objemového integrálu v jeho diskrétnej podobe (rovnica 2.15). Hlavnou myšlienkou tohto postupu je rozdeliť sčítanie a násobenie členov zo spomínanej rovnice do niekoľkých, ešte jednoduchších operácii, ktoré môžu byť vyhodnotené sekvenčne. Je možné použiť dva základné kompozičné prístupy.

Prvým z nich je kompozícia **spredu dozadu** (angl. front – to – back). Tento postup je aplikovaný v prípade, kedy sú svetelné paprsky (pohľadové paprsky, angl. viewing rays) vyslané smerom od pozorovateľa do konkrétneho vizualizovaného objemu. V predchádzajúcich častiach textu bolo spomenuté, že intenzitu, resp. radianciu v i - tom intervale je možno chápať ako farebný príspevok v podobe RGB. Nech je tento farebný príspevok konkrétneho intervalu domény označený ako C_i. Potom je možné pre iteratívny výpočet objemového integrálu pomocou tejto kompozičnej schémy použiť nasledovné rovnice:

$$\hat{C}_i = \hat{C}_{i+1} + \hat{T}_{i+1} C_i,$$
$$\hat{T}_i = \hat{T}_{i+1}(1 - \alpha_i),$$

pričom platí:

$$\hat{C}_n = C_n,$$
$$\hat{T}_n = 1 - \alpha_n$$

Výsledok konkrétneho intervalu, v tomto prípade iteračného kroku je reprezentovaný hodnotami $\hat{C}_i$ a $\hat{T}_i$. $\hat{C}_{i+1}$ a $\hat{T}_{i+1}$ sú akumulované výsledky z predchádzajúcich iteračných krokov. Hodnoty C_i a α_n sú odvodené na základe fyzikálneho popisu participujúceho média. Iterácia začína v bode $i = n$ najbližšie k pozorovateľovi a končí v bode $i = 0$ na zadnej strane objemu. V programátorskej praxi sa oplatí pre lepšie pochopenie odlišné pomenovanie daných premenných. Premenné $\hat{C}_i$ a $\hat{C}_{i+1}$ sa označia C_{dst}, premennú C_i je možno premenovať na C_{src}, $\hat{T}_i$ a $\hat{T}_{i+1}$ sa rovnajú $1 - \alpha_{dst}$ a nakoniec $\alpha_i = \alpha_{src}$. Potom pre front – to – back kompozičné schéma definitívne platí:

$$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst})C_{src}, \tag{2.18}$$

$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src} \tag{2.19}$$

Z predchádzajúcich rovníc je badateľná iteratívna povaha kompozície. Premenné s indexom src (skratka pre angl. slovo source) sú vstupné hodnoty v procese kompozície a reprezentujú hodnoty, resp. veličiny odvodené z optických materiálových vlastností participujúceho média. Naproti tomu, premenné s indexom dst (skratka pre angl. slovo destination) predstavujú akumulované hodnoty pre výslednú farbu a nepriehľadnosť. V procese kedy svetelný paprsok prechádza v scéne participujúcim médiom sú rovnice 2.18 a 2.19 opakovane vyhodnocované. Na konci tohto postupu je v premenných C_{dst} a α_{dst} konečná hodnota pre farbu a nepriehľadnosť.

V prípade, že obrátime smer postupu paprsku médiom, aplikujeme tzv. **back – to – front** kompozíciu. Ide o druhú možnosť, ako iteratívne vyhodnotiť objemový integrál. Iterácia začína v bode $i = 0$ objemu a končí v bode $i = n$. V tomto prípade však pre výpočet farebného príspevku $\hat{C}_i$ pre konkrétny i – ty interval nie je potrebná hodnota $\hat{T}_i$. Môže byť teda ignorovaná. Preto pre výslednú podobu rovníc pre back – to – front kompozíciu (po rovnakom premenovaní premenných ako v prípade predošlej kompozície) platí:

$$C_{dst} \leftarrow (1 - \alpha_{src})C_{dst} + C_{src} \tag{2.20}$$

Existujú aj iné kompozičné schémy. To však ale prevyšuje rámec tejto práce.

2.3. Rekonštrukcia a povaha objemových dát

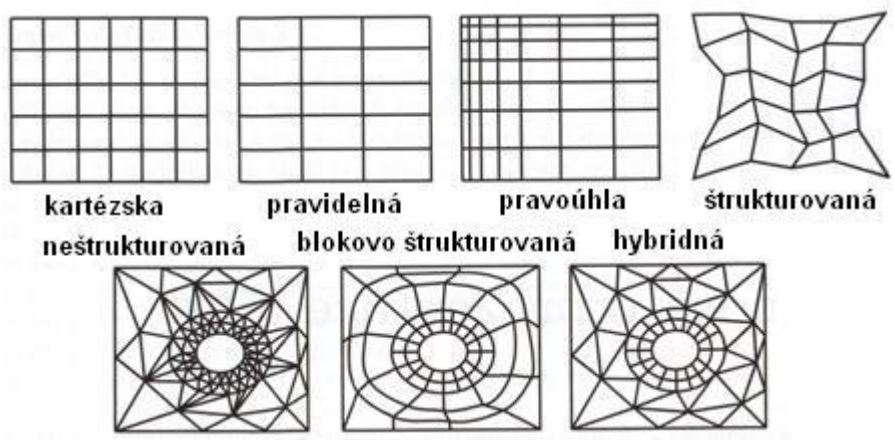
V úvodných podkapitolách tejto práce bol načrtnutý problém rekonštrukcie objemových dát. Všetky algoritmy slúžiace na vykresľovanie objemových dát majú za cieľ vizuálne extrahovať informáciu z trojrozmerného skalárneho priestoru, resp. trojrozmerných skalárnych dát[2]. Túto činnosť možno reprezentovať ako funkciu:

$$\emptyset: R^3 \rightarrow R$$

Ide teda o zobrazenie z trojrozmerného priestoru na skalárnu (číselnú) hodnotu. Objemové dáta sú však takmer vždy reprezentované pomocou diskrétnej mriežky. Presnejšie povedané, sú umiestnené na trojrozmernej diskrétnej mriežke v podobe vzorkou (angl. samples). Je tomu tak preto, lebo často je pôvod týchto dát z rôznych simulácií a meraní, kde je takáto reprezentácia najvhodnejšia. Diskrétna reprezentácia objemových dát však vedie k viacerým problémom. Ide najmä o problém rekonštrukcie funkcie $\emptyset$ vo všetkých bodoch trojrozmerného priestoru. Funkcia $\emptyset$ je totiž spojitá, je teda definovaná v ľubovoľnom bode trojrozmernej domény a nie iba v jednotlivých diskrétnych vzorkách. Rekonštrukcia tejto funkcie je náplňou podkapitoly 2.3.2. Vystáva takisto otázka, akým spôsobom sú vlastne pôvodné dáta reprezentované, resp. uložené na priestorových mriežkach. Tento problém je detailne vysvetlený v podkapitole 2.3.1.

2.3.1. Objemová reprezentácia telies, mriežky, voxel

U dát, akými sú aj objemové dáta hrá významnú úlohu tvar (doména) diskrétnej mriežky. Delenie geometrického tvaru mriežok navrhli Speray a Kennon, ktorí rozdelili obvyklé tvary do siedmych tried. Príklady jednotlivých typov mriežok sú uvedené na obrázku č. 14.

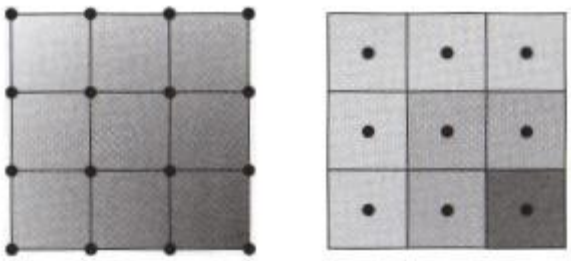


Obrázok č. 14: Príklady rôznych typov dátových mriežok

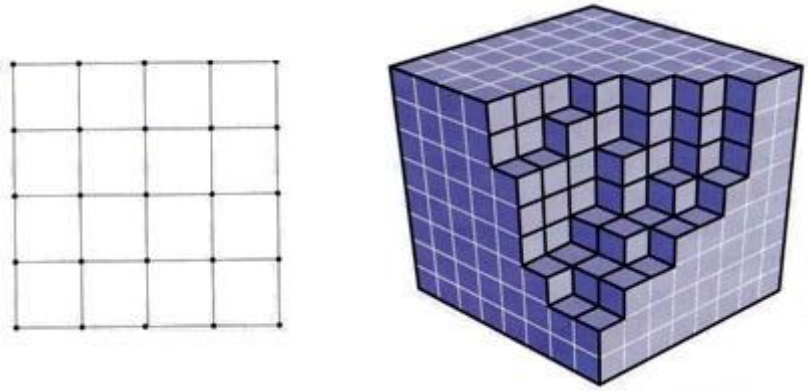
[2] Existuje viacero typov dát, nielen skalárne dáta; existujú napr. neskalárne dáta ako vektor, tenzor a pod.

Mriežka môže mať ľubovoľný počet rozmerov, ktorý sa označuje ako dimenzionalita domény. V prípade objemového vykresľovania sa najčastejšie vyskytujú trojrozmerné mriežky. Typ vzorkou informuje o počte hodnôt vo vzorku. Jedna hodnota sa označuje skalár, pole hodnôt vektor a matica hodnôt tenzor. Jedným z problémov diskrétnych objemových dát je, že je ich veľké množstvo. Tieto dáta kladú vysoké nároky na kapacitu pamäti a výkon CPU, resp. GPU. Vzhľadom k tomu, že namiesto spojitej informácie sa uchovávajú len diskrétne vzorky, sa objemové dáta ťažko otáčajú o uhly iné než pravé, tieto dáta sa problematicky zväčšujú alebo zmenšujú, prevzorkovaním sa stráca informácia o príslušnosti bunky k objektu a.i. Objemové dáta však majú niektoré výhody, ktoré spojito reprezentované objekty nemajú. Ide najmä o relatívne jednoduchú prácu s nameranými dátami, jednoduchá realizácia blokových operácií a logických operácií, spracovanie objemu dáta ako celku a nezávislosť na zložitosti scény (počte objektov v scéne) i na náročnosti generovania komplikovaných objektov. Pokiaľ ide o viacrozmerné dáta a dáta s neskalárnymi vzorkami (vektor, tenzor), nie je ich ukladanie žiaden problém. Problém je najmä orientácia v dátach a metódy zobrazovania a analýzy. V ďalších častiach textu sa pojednáva najmä o skalárnych dátach na pravidelnej priestorovej mriežke.

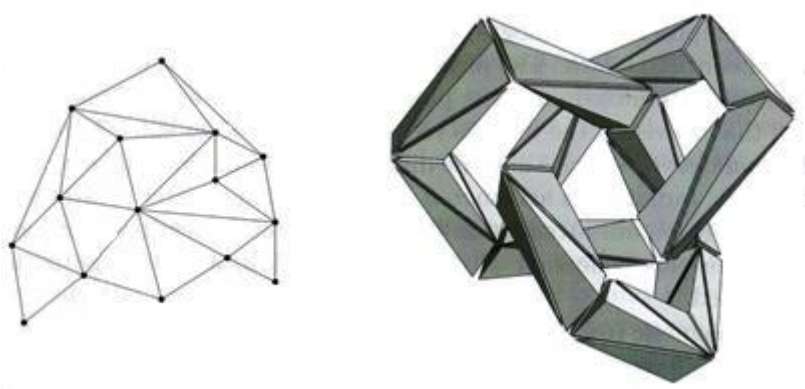
Základným objemovým elementom diskrétnej mriežky je **voxel.** Tento pojem vznikol ako analógia dvojrozmerného pixla. Inak povedané, voxel je najmenší element v trojrozmernom diskrétnom priestore. Často sa však v odbornej literatúre vyskytujú dve mierne odlišné definície voxela. Jedna interpretácia hovorí o tom, že voxel je malá kocka, resp. kváder, ktorý vypĺňa časť priestoru, resp. objemu a má priradenú určitú hodnotu (obrázok č. 15 vpravo). Iná interpretácia predpokladá, že voxely sú body v trojrozmernom priestore. Ide vlastne o body danej mriežky (bodové vzorky spojitého priestoru) spojené hranami, pričom vytvárajú v mriežke bunky (obrázok č. 15 vľavo). Hodnoty vo vnútri buniek sa vypočítavajú lineárnou alebo kvadratickou interpoláciou hodnôt vo voxeloch tvoriacich danú bunku. Priestorová bunka môže mať tvar kocky (pravidelná mriežka – obrázok č. 15, 16 vpravo), štvorstenu (obrázok č. 17 vpravo) alebo dokonca n – stenu (neštrukturovaná mriežka).



Obrázok č. 15: Základné objemové elementy (body mriežky vs. kocky)



Obrázok č. 16: Pravidelné mriežky v 2D (štvorcové bunky) vs. 3D (bunky v podobe kociek)



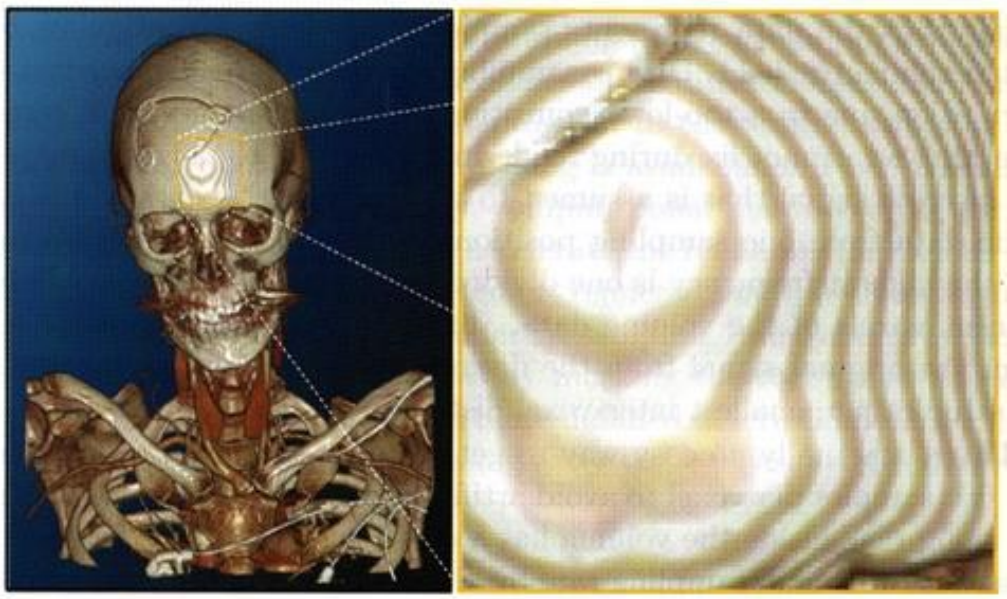
Obrázok č. 17: Neštruktúrované mriežky s bunkami v tvare trojuholníkov a štvorstenov

Pravidelné n - dimenzionálne mriežky majú výhodu vo svojej pravidelnej štruktúre, čo vedie k ich kompaktnej reprezentácii v pamäti počítača (v podobe n - rozmerných polí) a v rýchlom prístupe k jednotlivým dátovým bunkám. Nie sú však príliš flexibilné. Práve kvôli tomu sa často využívajú mriežky s inou štruktúrou a tým pádom s vyššou úrovňou flexibility.

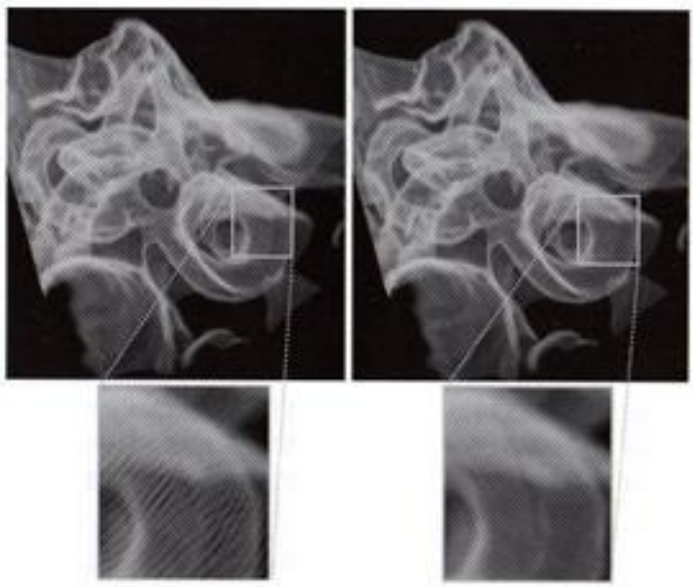
2.3.2. Rekonštrukcia objemových dát

Objemové dáta sú reprezentované v diskrétnej podobe – typicky na pravidelnej mriežke alebo mriežke s bunkami v tvare štvorstenov. Takáto diskretizácia vedie k úlohe rekonštrukcie skalárnej funkcie ∅ vo všetkých bodoch trojrozmernej domény (viď. problém výpočtu hodnôt vo vnútri buniek pomocou interpolácie z predošlej kapitoly).

Dôležitou otázkou pre správnu rekonštrukciu danej spojitej funkcie je minimálny počet vzorkou rozprestretých na mriežke. Tento počet upravuje tzv. Nyquistov teorém. Ten hovorí o tom, že ak chceme zrekonštruovať pôvodný spojitý signál z diskrétnej verzie toho istého signálu, musí byť vzorkovacia frekvencia minimálne dvakrát väčšia než hodnota najväčšej frekvencie vo vstupnom signáli. V opačnom prípade dôjde k efektu tzv. aliasingu – spojitý signál sa z jeho diskrétnej podoby zrekonštruuje nesprávne (z dôvodu nedostatočného počtu vzorkou v objemových dátach). Existuje viacero techník, ako nežiadané artefakty z výstupného obrazu odstrániť. Ide napr. o metódu adaptívneho vzorkovania, kedy sa počet vzorkou zvyšuje podľa potreby. Častou metódou potlačujúcou aliasing je aj tzv. trasenie paprskov (angl. jittering). Na nasledujúcich obrázkoch je možné vidieť, aké artefakty vo výstupnom obraze spôsobuje nesprávne vzorkovanie, resp. nesprávny počet vzorkou v diskrétnom signáli (objemových dátach).



Obrázok č. 18: Vznik aliasingu v dôsledku nízkeho počtu vzorkou v objemových dátach



Obrázok č. 19: Objemové vykreslenie vnútorného ucha s nízkym (vľavo) a vysokým (vpravo) počtom vzorkou

Obrázok č. 20: Objemové vykresľovanie s (vľavo) a bez (vpravo) techniky trasenia paprskov

Správne vzorkovanie objemových (počet vzorkou v objemových dátach) je teda dôležitým predpokladom ako zabrániť nežiadaným artefaktom v obraze. Vzchádzajúc z Nyquistovho teorému musia byť splnené viaceré požiadavky na objemové dáta, resp. diskrétny signál. V spojitom a periodickom vstupnom signáli reprezentovanom funkciou $f(t)$ určíme jeho najvyššiu frekvenciu ϑ_f. Maximálna frekvencia znamená, že Fourierova transformácia signálu $f(t)$ je nulová mimo frekvenčný interval $\langle -\vartheta_f | \vartheta_f \rangle$. Potom pre hľadanú vzorkovaciu frekvenciu (Nyquistovu frekvenciu) platí $\vartheta_n = 2\vartheta_f$. Z toho vyplýva, že pre správne vzorkovanie potlačujúce aliasing sa musí na jednotkovej vzdialenosti v objemových dátach nachádzať viac než $2\vartheta_f$ vzorkou. Pre rovnomerné vzorkovanie na frekvencii $\vartheta_s > \vartheta_n$ je teda možné jednotlivé vzorky, resp. ich pozíciu určiť funkciou $f_i = f(i/\vartheta_s), i \in N$.

Pôvodný spojitý signál je možné z takéhoto diskrétneho signálu získať pomocou vzorkou f_i nasledovne:

$$f(t) = \sum_i f_i \,\mathrm{sinc}\big(\pi(\vartheta_s t - i)\big) \tag{2.21}$$

Funkcia $sinc$ (sinus cardinalis) je definovaná:

$$\mathrm{sinc}(t) = \begin{cases} \frac{\sin(t)}{t} & \text{if } t \neq 0 \\ 1 & \text{if } t = 0 \end{cases}$$

Signál, ktorého Fourierova transformácia je nulová mimo frekvenčného intervalu $\langle -\vartheta_f | \vartheta_f \rangle$ sa nazýva signál s obmedzeným pásmom (v tomto prípade s obmedzenou frekvenciou). Avšak v praxi je často frekvenčný obsah vstupných dát neznámy. Z toho dôvodu je problematické určiť správne vzorkovanie takýchto dát. V tomto prípade sa nasadzujú nízko priepustné filtre s cieľom obmedziť maximálnu frekvenciu na kontrolovanú a vopred známu hodnotu.

Rovnica 2.21 pre definíciu spojitého signálu je príkladom konvolúcie. Vo všeobecnosti ma konvolúcia nasledujúci tvar:

$$g(m) = (f * h)(m) = \sum_i f(i)h(m - i)$$

Ide o konvolúciu dvoch diskrétnych funkcií f a h. Konvolúcia úzko súvisí s filtráciou signálov. Presnejšie, f je vstupný signál, h je filtračná maska (filter) a funkcia g odfiltrovaný výstup. Z toho vyplýva, že rovnica 2.21 je konvolúciou diskrétneho signálu f_i s filtračnou maskou $sinc$. Táto filtračná maska má však viacero problémov. Hodnoty funkcie $sinc$ oscilujú okolo 0 pozdĺž celej funkčnej domény. Z toho dôvodu, musí byť konvolúcia vyčíslená pre všetky vstupné hodnoty f_i, čo je však časovo náročné. Nasadzujú sa preto iné rekonštrukčné filtre než funkcia sinus cardinalis. Je možné použiť „box“ filter, ktorý vedie k tzv. nearest – neighbor interpolácii ako spôsobu rekonštrukcie pôvodného signálu, resp. dát. Pri tomto type interpolácie dát je hodnota rekonštruovanej funkcie v danom bode objemu nastavená na hodnotu najbližšieho susedného vzorku. Ide však o interpoláciu nie príliš presnú. Je teda vhodné použiť iné rekonštrukčné filtre. Pre potreby objemového vykresľovania sa hodí nasledujúci filter, resp. filtračná maska:

$$h(x, y, z) = h_x(x)h_y(y)h_z(z)$$

pričom $h_x(x)$, $h_y(y)$, $h_z(z)$ sú filtre fungujúce v jednej doméne, postupne v smeroch x, y a z.

Tento typ rekonštrukčného filtra vedie k lineárnej interpolácii. Presnejšie povedané, rozlišuje interpolácie pozdĺž všetkých troch dimenzií a preto umožňuje vypočítať rekonštruovanú hodnotu vo vnútri bunky dátovej mriežky ako sekvenciou lineárnych interpolácií vo všetkých troch smeroch trojrozmerného priestoru.

Lineárna interpolácia medzi dvomi bodmi a, b sa určí nasledovne:

$$f(p) = (1 - x)f(a) + xf(b),$$

pričom $f(a)$ a $f(b)$ sú funkčné hodnoty vo vzorkoch a, b.

Lineárna interpolácia v dvoch dimenziách sa nazýva bilineárna interpolácia. Bilineárna interpolácia v bode p je vlastne sekvenciou po sebe idúcich dvoch lineárnych interpolácií:

$$f(p) = (1 - y)f(p_{ab}) + yf(p_{cd}),$$

využívajúc medzivýsledky z lineárnej interpolácie pozdĺž osi x:

$$f(p_{ab}) = (1 - x)f(a) + xf(b),$$
$$f(p_{cd}) = (1 - x)f(d) + xf(c)$$

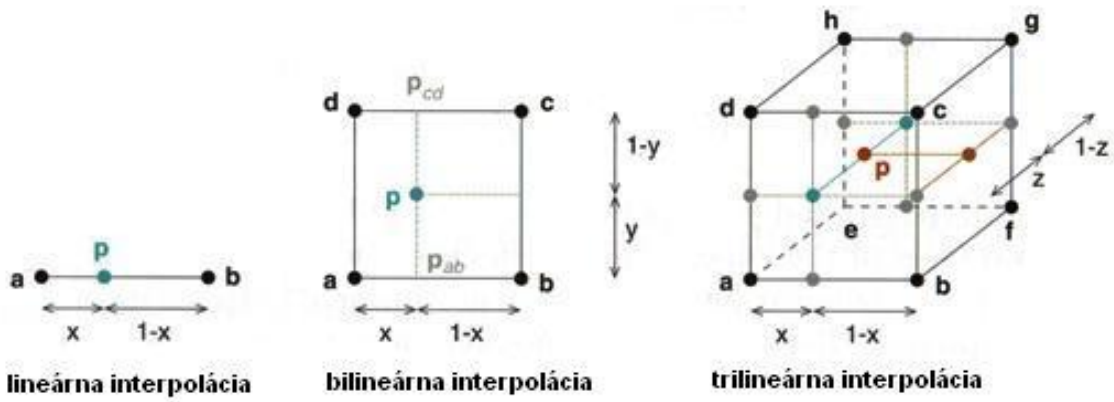
Kombináciou predchádzajúcich troch rovníc pre bilineárnu interpoláciu platí:

$$f(p) = (1 - x)(1 - y)f(a) + (1 - x)yf(d) + x(1 - y)f(b) + xyf(c)$$

Trilineárnu interpoláciu v troch dimenziách je možné vypočítať lineárnou interpoláciou medzi dvomi medzivýsledkami získanými z bilineárnej interpolácie:

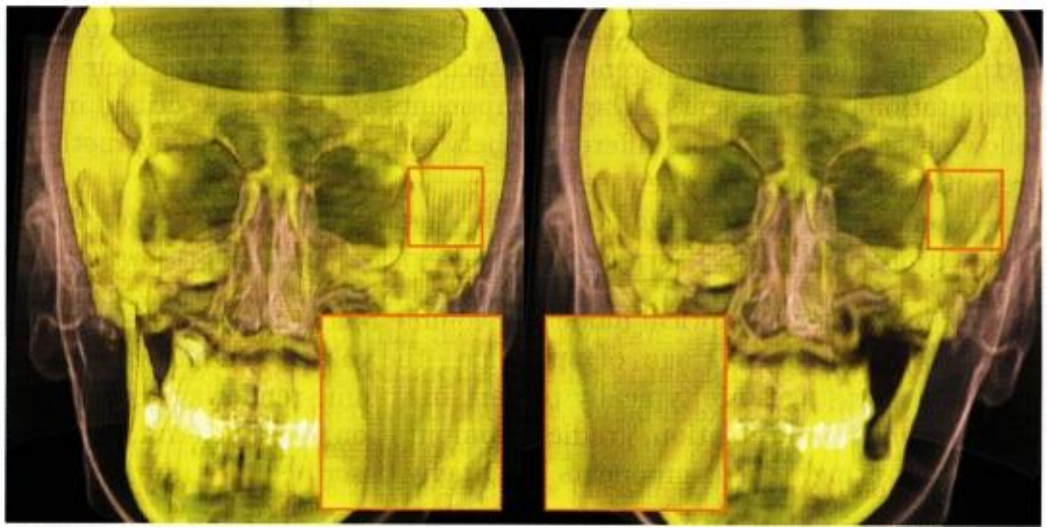
$$f(p) = (1 - z)f(p_{abcd}) + zf(p_{efgh})$$

Členy $f(p_{abcd})$ a $f(p_{efgh})$ sa získajú bilineárnou interpoláciou vo vnútri dvoch strán bunky dátovej mriežky, v ktorej danú hodnotu chceme získať. Pre názorné pochopenie je možné na obrázku č. 21 vidieť všetky tri typy interpolácií.



Obrázok č. 21: Tri typy interpolácii často sa vyskytujúcich v objemovom vykresľovaní

Všetky spomenuté typy interpolácií zohrávajú v oblasti objemového vykresľovania významnú úlohu. Sú totiž rýchlo vyčísliteľné a to i vďaka hardwarovej podpore moderných GPU. Existujú však aplikácie, kedy nepostačuje kvalita výsledných snímok a nasadzujú sa iné filtračné, resp. rekonštrukčné metódy. Na nasledujúcom obrázku je možné sledovať, že i trilineárna interpolácia môže spôsobovať významné artefakty v obraze.



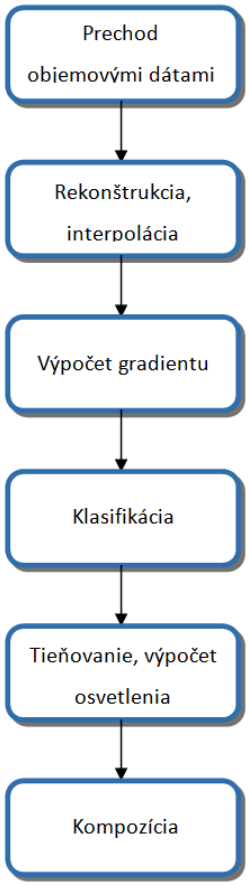
Obrázok č. 22: Porovnanie trilineárnej a kubickej B – spline interpolácie

3. Vizualizácia objemových dát

V súčasnosti existuje viacero algoritmov zobrazujúcich objemové dáta. Ich spoločnou úlohou je vypočítať, resp. vyhodnotiť daný optický model (najčastejšie emisno – absorpčný model, kapitola 2.1.2) pomocou diskrétnej aproximácie (kapitola 2.2) objemového integrálu (kapitola 2.1.3). Podľa rôznych kritérií, spôsobu realizácie a funkčnosti týchto algoritmov sa dajú rozdeliť do viacerých skupín. Každý z nich pozostáva z viacerých komponent, ktoré na seba nadväzujú, pričom každá komponenta rieši určitú časť problému súvisiaceho s vykresľovaním objemových dát. Takáto postupnosť komponent, resp. častí algoritmu vedúcich ku konečnému výstupu sa nazýva volume – rendering pipeline[3]. Náplňou tejto kapitoly bude práve rozobrať samotnú volume – rendering pipeline a jednotlivé algoritmy (niektoré z nich len v stručnosti).

3.1. Volume – rendering pipeline

Vysvetlenie, čo vlastne volume – rendering pipeline (ďalej už len VRP) znamená, bolo podané v úvode celej tejto kapitoly. Na tomto mieste je vhodné podrobne vysvetliť, z akých komponent vlastne VRP pozostáva a aká je náplň ich činnosti. Štruktúra VRP je na nasledujúcom obrázku:



Obrázok č. 23. Komponenty volume – rendering pipeline

[3] Je problematické preložiť pojem volume – rendering pipeline do slovenčiny, preto som sa o to ani nepokúšal

Prechod objemovými dátami:

Výber vzorkovacích pozícií pri prechode objemovými dátami. Napr., metóda Ray Casting vysiela do scény paprsky, tie prechádzajú objemov s dátami a v určitých bodoch (na pozíciach vzorkou) sa vyčísli hodnota danej veličiny popísanej objemovými dátami. Tieto vzorky slúžia ako vstup pre diskretizáciu, resp. vyčíslenie spojitého objemového integrálu.

Interpolácia:

Pozície vzorkou z predchádzajúceho kroku VRP sa obvykle líšia od pozície bodov na diskrétnej objemovej dátovej mriežke. Všetky algoritmy slúžiace na vykresľovanie objemových dát majú za cieľ vizuálne extrahovať informáciu z trojrozmerného skalárneho priestoru, resp. trojrozmerných skalárnych dát (z objemových dát). Túto činnosť možno reprezentovať už známou funkciou:

$$\emptyset: R^3 \rightarrow R$$

Objemové algoritmy potrebujú vyčísliť hodnotu danej skalárnej veličiny na pozíciach vzorkou z predošlého kroku VRP. Ako bolo ale už spomenuté, pozícia vzorkou je odlišná od pozície bodov v trojrozmernom skalárnom poli (na dátových mriežkach). Tento stav vedie k úlohe interpolácie hodnôt funkcie $\emptyset$ na pozíciach vzorkov pomocou hodnôt rozprestretých na objemových dátových mriežkach.

Výpočet gradientu:

Tradičné osvetľovacie modely ako napr. Phongov osvetľovací model sú založené na znalosti normálových vektorov v jednotlivých bodoch na povrchu zobrazovaného telesa. Popisujú lokálnu orientáciu určitej časti povrchu – plochy. S cieľom prispôsobiť tieto lokálne osvetľovacie modely potrebám objemového vykresľovania, predpokladá sa, že sa svetlo z externých zdrojov v scéne odráža na tzv. izoplochách vo vnútri objemových dát. Každý bod p vo vnútri objemu ma asociovanú skalárnu hodnotu[4] $f(p)$. Izoplochou $I(p)$ prechádzajúcou bodom p sa chápe množina všetkých takých bodov objemu, ktoré majú priradenú tú istú skalárnu hodnotu $f(p)$, t.j.:

$$I(p) = \{x | f(x) = f(p)\}$$

Izoplocha je teda podmnožina konkrétneho objemu. Avšak okrem homogénnych oblastí a singulárnych bodov v objeme tvoria izoplochy „skutočný“ povrch. Normálový vektor, ktorý sa použije pre tieňovanie bodu v objeme je teda jednotkový vektor kolmý vzhľadom k izoploche prechádzajúcej týmto bodom.

[4] V texte sa predpokladá, že bod v objeme má priradený vždy skalár, môže to však byť vektor ale i tenzor

Gradient skalárneho poľa (objemových dát) v bode x objemu je definovaný nasledovne:

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x} \\ \frac{\partial f(x)}{\partial y} \\ \frac{\partial f(x)}{\partial z} \end{pmatrix}$$

Ide o vektor vyjadrujúci smer a veľkosť najväčšej zmeny skúmaného skalárneho poľa v bode x. Tento smer je vždy kolmý vzhľadom k izoploche. Z toho vyplýva, že normálový vektor potrebný pre správne vyhodnotenie osvetľovacie modelu v bode p objemu je možné získať ako gradient v tom istom bode. Je však potrebné daný gradient normalizovať. Osvetľovacie modely totiž pracujú s jednotkovými normálovými vektormi. Existuje viacero techník ako určiť gradient v objemových dátach – to však prevyšuje rámec tejto práce.

Klasifikácia:

Objemová dáta sú abstraktné, najčastejšie skalárne dáta reprezentujúce, resp. popisujúce nejakú veličinu, ktorá sa mení v závislosti na polohe v priestore. Pre potreby vizualizácie takýchto dát je potrebné vyčísliť objemový integrál. Na to je ale nutné disponovať v každom bode objemu emisným a absorpčným koeficientom, definovanými v predchádzajúcej kapitole. Neexistuje však prirodzený spôsob, ako z objemových dát získať dané koeficienty. Práve preto užívateľ danej aplikácie musí rozhodnúť ako jednotlivé vnútorné štruktúry v objemových dátach vyzerajú pomocou tzv. transfer functions. Ide o proces, v ktorom užívateľ mapuje jednotlivým hodnotám objemových dát konkrétne emisné a absorpčné koeficienty v podobe farby C a nepriehľadnosti α. Proces hľadanie tej správnej transfer function, t.j. toho správneho mapovania dát a koeficientov sa nazýva klasifikácia. Práve klasifikácia umožňuje zobraziť v objeme jednotlivé materiály, súčasti alebo štruktúry.

Tieňovanie, výpočet osvetlenia:

Úzko súvisí s výpočtom gradientu. Je možné použiť rôzne osvetľovacie modely, ako napr. Phongov osvetľovací model, Blinn – Phong osvetľovací model, Cook – Torrance osvetľovací model a.i. Aby bolo možné vyhodnotiť osvetľovací model je potrebné rozšíriť zdrojový člen q zavedený v kapitole 2.1.1 nasledovne:

$$q_{rozšírený} = q_{emisia}(x, w) + q_{osvetlenie}(x, w)$$

Člen q_{emisia} je identický zdrojovému členu q emisno – absorpčného modelu z kapitoly 2.1.1. Doplnený člen $q_{osvetlenie}$ popisuje dodatočnú svetelnú energiu pochádzajúcu z odrazov v scéne.

Kompozícia:
Kompozícia bola podrobne vysvetlená v kapitole 2.2.2.

Komponenty akými sú interpolácia, výpočet gradientov, tieňovanie a klasifikácia fungujú na lokálnej báze – sú realizované v blízkosti vzorkou alebo priamo na ich pozíciach. Práve preto sú tieto komponenty VRP nezávislé na zvolenej metóde zobrazovania objemových dát a môžu byť opätovne použité v rôznych algoritmoch.

3.2. Algoritmy vykresľujúce objemové dáta

Algoritmy vykresľujúce objemové dáta je možné rozdeliť podľa niekoľkých kritérií na viacero skupín. Jedným z takýchto kritérií je súradnicová sústava, v akej daný algoritmus pracuje. Na tomto základe sa rozlišujú algoritmy fungujúce v:

- súradnicovej sústavy obrazovky (angl. image order methods)
  - ray casting
- súradnicovej sústave objektu (angl. object order methods)
  - texture slicing
  - splatting
  - shear – warp volume rendering[5]

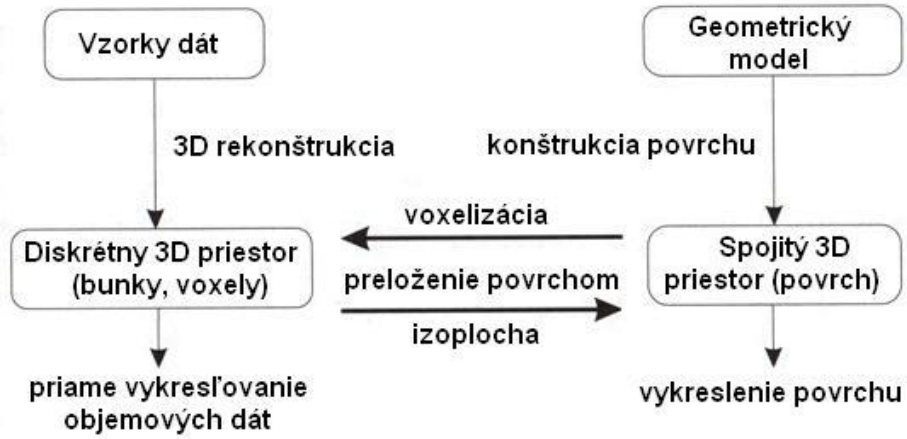
Metódy z prvej skupiny algoritmov pracujú na princípe, kedy každý pixel na obrazovke, t.j. v súradnej sústave obrazovky, slúži ako štartovací bod pre vyhodnotenie objemového integrálu. Typickým zástupcom je algoritmus známy ako Ray Casting (slov. vrhanie paprskov). Tomuto algoritmu bude venovaná samostatná podkapitola. Algoritmy pracujúce v súradnej sústave objektu, v tomto prípade objemových dát, prechádzajú objemom na základe nejakej vnútornej štruktúry. Takéto prejdené časti objemu sa postupne premietnu na obrazovku. Typickým príkladom je metóda nazvaná texture slicing.

Ďalšie možné delenie objemových algoritmov je nasledovné:

- algoritmy zobrazujúce povrchy
- priame objemové algoritmy

Porovnanie oboch prístupov je možné sledovať na nasledujúcom obrázku:

[5] Prekladať názvy algoritmov do slovenčiny je problematické, preto som ich uviedol v ich anglickom originály



Obrázok č. 24: Metódy zobrazovania objemových dát

Algoritmy zobrazujúce povrchy (angl. surface – fitting algorithms) vytvárajú najskôr geometrickú reprezentáciu povrchu (obrázok č. 24 vpravo). Pracujú preto s objemovými dátami nepriamo. Najskôr ich preložia povrchom, ktorý reprezentujú geometrickými primitívami – najčastejšie trojuholníkovými sieťami (angl. triangle mesh). Až následne tieto primitíva zobrazujú klasickými metódami počítačovej grafiky. Výhodou je, že sa prechodom k ploškovej reprezentácii značne zníži množstvo dát. Namiesto celej mriežky vzorkou zostanú iba povrchové elementy špecifikované výrazne nižším počtom hodnôt. Povrchové siete trojuholníkov je možné vykresľovať bežnými algoritmami s viditeľnosťou a hladkým tieňovaním alebo metódou sledovania paprsku (angl. ray tracing).

Priame objemové algoritmy (angl. direct volume rendering) zobrazujú skalárne dáta priamo, bez nutnosti prevodu do povrchovej reprezentácie (obrázok č. 24 vľavo). Využívajú plnú priestorovú informáciu a pred zobrazením nie je potrebné vedieť, či vzorok (voxel alebo vrchol bunky – záleží na interpretácii) patrí alebo nepatrí k zobrazovanému objektu, resp. jeho povrchu. Algoritmy tejto skupiny je možné rozšíriť i na zobrazovanie polopriehľadných materiálov a štruktúr vo vnútri iných štruktúr v rámci objemových dát. Umožňujú súčasne vykresľovať tak rozhranie medzi materiálmi ako i ich vnútrajšok.

Vychádzajúc z obrázku č. 24 je jasné, že k prevodu spojitej reprezentácie na objemovú je možné použiť techniku tzv. voxelizácie, t.j. navzorkovanie spojitého povrchu (geometrického popisu) v určitom rozlíšení[6]. Často sa spolu s hodnotou vzorku ukladá i hodnota normály povrchu, ktorá je potrebná pri vyhodnotení osvetľovacieho modelu. Naopak, k prevodu objemovej reprezentácie na povrchovú poslúži technika, o ktorej bola reč v predchádzajúcej podkapitole – hľadanie, resp. preloženie izoplochou.

[6] Technika voxelizácie je mimo iné súčasťou mojej vyvinutej aplikácie

3.2.1. Algoritmy zobrazujúce povrchy

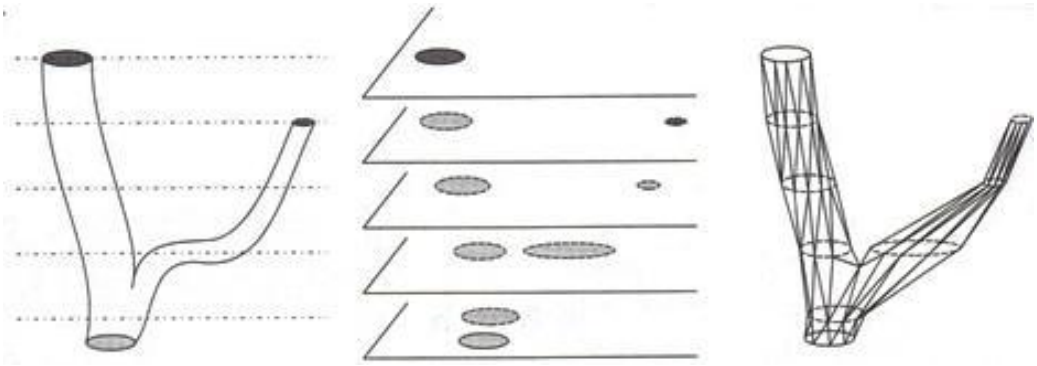
Algoritmy, ktoré pri vizualizácii objemových dáta zobrazujú povrchy, pracujú nepriamo, pretože samotnému zobrazovaniu predchádza generovanie (pomocnej – proxy geometry) povrchovej reprezentácie.

V prípade, že sa zobrazuje priebeh konkrétnej fyzikálnej veličiny (teplota, tlak, erózia a pod.) pomocou plôch, určí sa zobrazovaný povrch ako miesto, kde prebieha plocha s konštantnou hodnotou danej veličiny (viď. definícia izoplôch v kapitole 3.1).

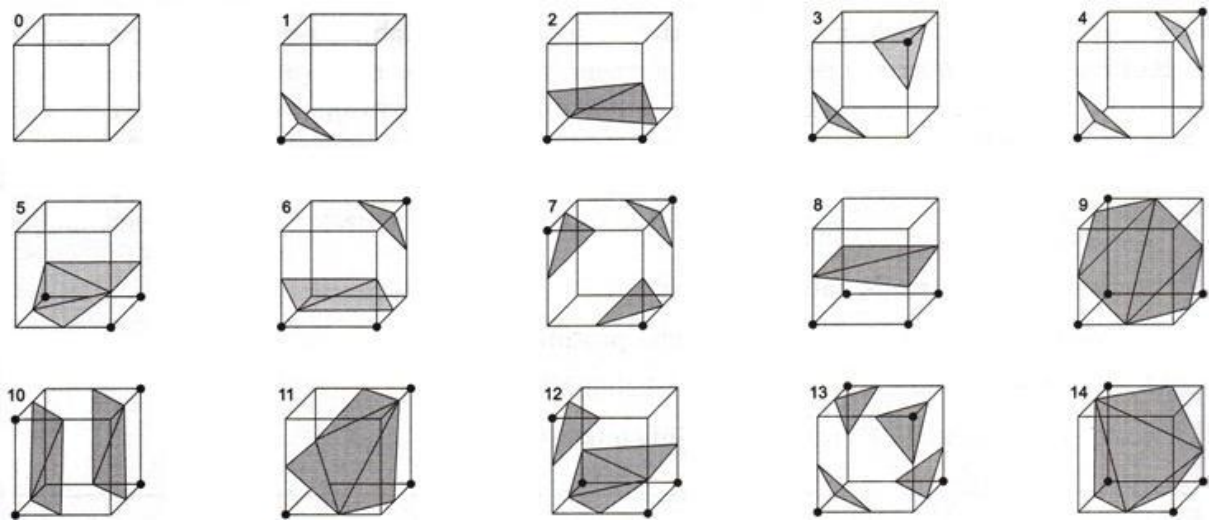
Pri vykresľovaní skutočných útvarov, ako napr. ľudských orgánov z dát nameraných pomocou CT alebo MRI, sa jedná o obraz existujúcich objektov. O príslušnosti voxelu k zobrazovanému objektu je možné rozhodnúť v procese klasifikácie VRP. Tento proces si vyžaduje do značnej miery znalosť užívateľa danej aplikácie. Ten musí vedieť, s akými dátami pracuje, aké vnútorné štruktúry sa v objemových dátach nachádzajú a musí vedieť rozpoznať hranice, resp. prechod medzi jednotlivými štruktúrami. K dispozícii sú dve možnosti ako rozhodnúť o príslušnosti voxelu k objektu pomocou klasifikácie. Prvou z nich je, že sa zvolí nejaké kritérium, pomocou ktorého je možné rozpoznať hranicu počas vykresľovania samotných dát (napr. kritérium pre výber izoplochy). Druhý možný prístup je ten, že pred samotné zobrazovanie dát sa zaradí krok, ktorého výsledkom bude označenie, ktorý voxel náleží, a ktorý nenáleží oplášťovanému objektu. Takýmto spôsobom vznikne binárny objem slúžiaci ako priestorová maska, ktorá vymedzuje časti priestorových dát, ktoré náležia zobrazovanému objektu. Takto vymedzené dáta sú následne predmetom samotného vykresľovania. Jednoduchším spôsobom, ako vytvoriť takýto binárny objem je technika prahovania.

Existuje viacero algoritmov generovania povrchovej reprezentácie v miestach izoplôch:

- **prepojovanie kontúr** – dáta sa spracovávajú postupne po vrstvách (dvojice rezov), v jednotlivých rezoch sa najskôr označia obrysy hranice a následne sa v každej dvojici rezov stanoví topológia kontúr (určenie čo sa bude s čím spojovať) a prevedie sa opláštenie (angl. tessellation)
- **povrchové kocky** – algoritmus chápe objemové elementy ako plné kocky, po nájdení povrchových kociek (sledovaním hranice okolo zadanej hodnoty) sa každá z nich prevedie na šesticu štvoruholníkov, ktoré sa následne vykreslia
- **pochodujúce kocky (**angl. **marching cubes)** – najčastejšie nasadzovaný algoritmus pre tvorbu sietí trojuholníkov, ktorá aproximuje povrch prechádzajúci jednotlivými objemovými elementami (v tomto prípade kockami)
- **rozdelenie kociek (**angl. **dividing cubes)** – problém pomalej rasterizácie obrovského množstva malých plôch sa vyriešil tým, že tento krok je vynechaný a namiesto plôch sa generujú povrchové body s normálou



Obrázok č. 25: Algoritmus prepojovania kontúr. Priestorový objekt, kontúry v rovnobežných rezoch a rekonštrukcia povrchu opláštením sieťou trojuholníkov



Obrázok č. 26: Základné prípady konfigurácie vrcholov kocky v algoritme pochodujúcej kocky a spôsob, akým povrch objektu pretína kocku

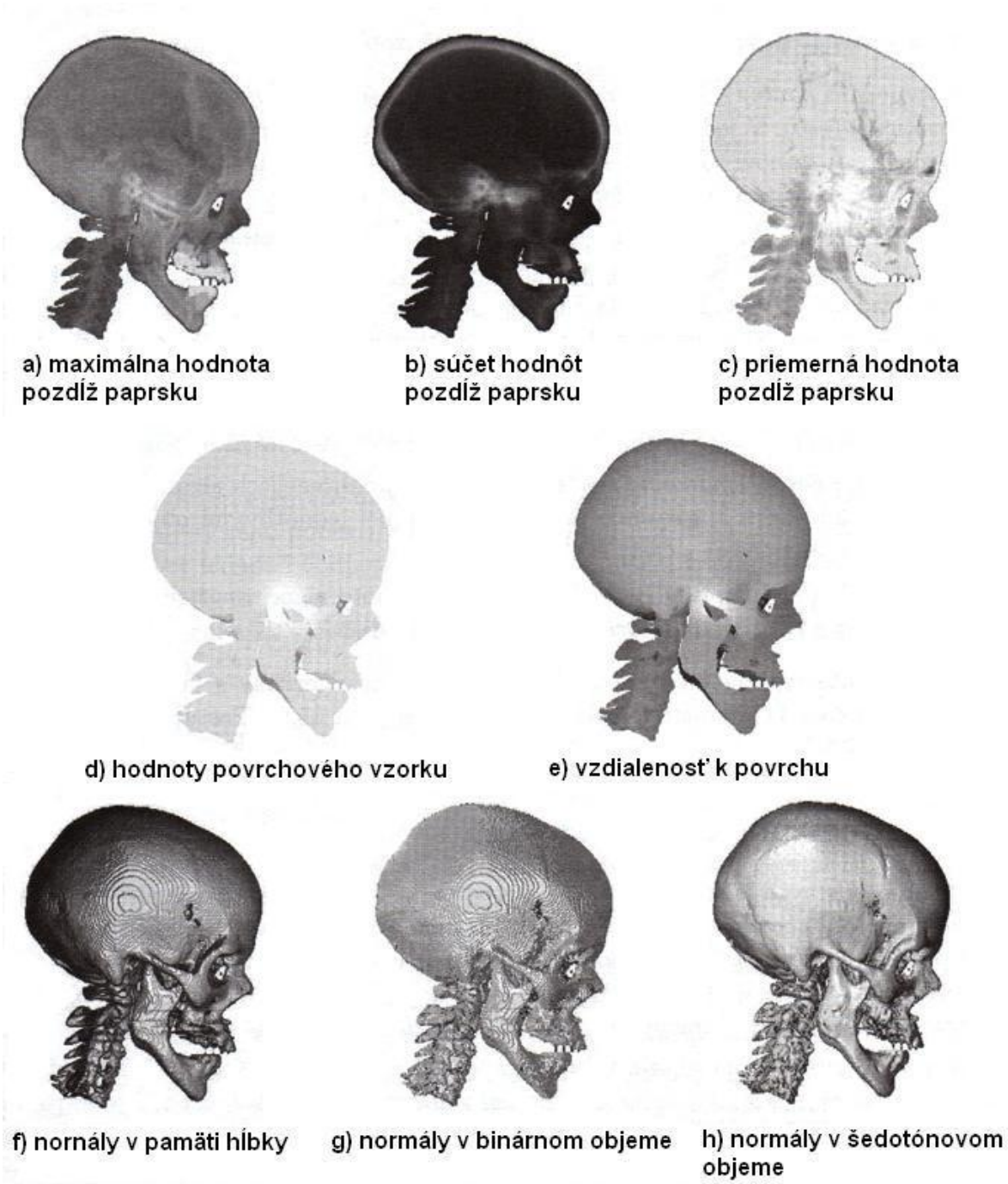
3.2.2. Priame zobrazovanie objemu

Priame objemové algoritmy zobrazujú skalárne dáta priamo, bez nutnosti prevodu do povrchovej reprezentácie. Využívajú plnú priestorovú informáciu a pred zobrazením nie je potrebné vedieť, či voxel patrí alebo nepatrí k zobrazovanému objektu, resp. jeho povrchu. Ide o veľmi často sa vyskytujúce metódy zobrazovania objemových dát. Typickými zástupcami tejto skupiny algoritmov je metóda vrhania paprskov (angl. Ray Casting) a algoritmus Texture Slicing. Ich podrobná charakteristika bude náplňou ďalších podkapitol.

Na tomto mieste je vhodné uviesť, že zobrazovanie objemových dát metódou vrhania paprsku je možné rozdeliť podľa toho, s akými dátami sa pracuje a čo z nich sa zobrazuje, na:

- metódy pracujúce s dátami bez hľadania povrchu (bez segmentácie)
- metódy hľadajúce povrch, avšak nezisťujúce normálu
- metódy hľadajúce povrch so zisťovaním normály (odhadom normály)

Na nasledujúcom obrázku sú uvedené ukážky získané aplikáciou troch uvedených skupín metód na rovnaké dáta. Ide o priestorové dáta ľudskej lebky z CT (341 rezov, každý rez ma rozlíšenie 226 x 272 vzorkou).



Obrázok č. 27: Varianty metódy vrhania paprskov (angl. Ray Casting)

Bod a) na obrázku č. 27 zodpovedá algoritmu označovanému ako *maximum intensity projection*, t.j. zobrazuje iba najjasnejšie štruktúry pozdĺž paprsku:

$$I = \max_{i \in J}(I_i),$$

I – výsledná intenzita jasu pixla pre jediný vrhnutý paprsok

I_i – skalárna hodnota intenzity i – teho vzorku pozdĺž paprsku

J - množina započítaných vzorkov (vzorky zasiahnuté paprskom a vyhovujúce danému kritériu)

Inou možnosťou (angl. *summed intensity projection*) je vypočítať jas pixla ako súčet tých intenzít pozdĺž paprsku, ktoré ležia v zadanom intervale (bod b) na obrázku č. 27):

$$I = \sum_{i \in J}(I_i)$$

Modifikáciou predchádzajúcej metódy je možné získať metódu, ktorá normalizuje príspevky rôzneho počtu rovnakých intenzít (angl. *average intensity projection*). Výsledný jas pixla sa rovná priemeru intenzít pozdĺž paprsku, ktoré ležia v zadanom intervale (bod c) na obrázku č. 27):

$$I = \frac{\sum_{i \in J}(I_i)}{|J|}$$

Bod d) na obrázku č. 27 zodpovedá technike (angl. *voxel value projection*), ktorá nezobrazuje tvar, ale iba farbu objektu. Ide v podstate o obdobu konštantného tieňovania v prípade telies zadaných pomocou hraničnej reprezentácie. V objeme sú k dispozícii vzorky patriace k povrchu objektu. Vopred sú takisto známe hodnoty intenzity jasu vo všetkých vzorkoch. Tento algoritmus nastaví výslednú intenzitu pixla, do ktorého sa premietol vzorok prislúchajúci k povrchu, na hodnotu intenzity jasu takto premietnutého vzorku, t.j.:

$$I = I_s,$$

I_s - intenzita jasu vzorkou patriacich k povrchu

Ďalšia technika (angl. *depth/distance only*) nastavuje jas pixla na obrazovke podľa hodnoty vzdialenosti najbližšieho vzorka na dráhe paprsku, t.j. nastavuje intenzitu jasu pixla $P = [p_0, p_1]$ na obrazovke podľa hodnoty z v pamäti hĺbky (angl. depth – buffer):

$$I = Z(I_s) = Z(p_0, p_1)$$

Táto metóda potlačuje šum vzorkovania, ale spolu s ním aj hrany a nespojitosti, ktoré sú dôležité pri vnímaní tvarov človekom (bod e) na obrázku č. 27).

Kvalitnejšieho znázornenia povrchu, oproti algoritmom z predchádzajúcich odstavcov je možné dosiahnuť technikami, ktoré sa pokúšajú odhadnúť orientáciu povrchu v mieste dopadu paprsku. V tomto bode sa následne vypočíta osvetlenie napr. podľa Phongovho osvetľovacieho modelu. Tieto algoritmy pracujú s binárnym objemom (je možné určiť, ktoré voxely, resp. vzorky prislúchajú k povrchu alebo objektu) a s objemom vzorkou (každý vzorok má pridelenú intenzitu jasu).

Bod f) na obrázku č. 27 prislúcha technike (angl. *Z – buffer gradient shading*), kedy sa normála k povrchovému vzorku aproximuje vektorom, ktorého kolmým priemetom do plochy obrazovky je gradient v pamäti hĺbky. Táto metóda zaznamenáva tvar telesa, ale na zaoblených povrchoch vytvára viditeľné artefakty v podobe vrstevníc.

Ďalší algoritmus (angl. *voxel gradient shading*) odhaduje orientáciu povrchu podľa gradientu v binárnom objeme vzorkou. Výsledné normály nadobúdajú iba jednu z 27 možných hodnôt (bod g) na obrázku č. 27). Podobne ako predchádzajúca metóda, i v tomto prípade vznikajú na zaoblených povrchoch artefakty v podobe vrstevníc.

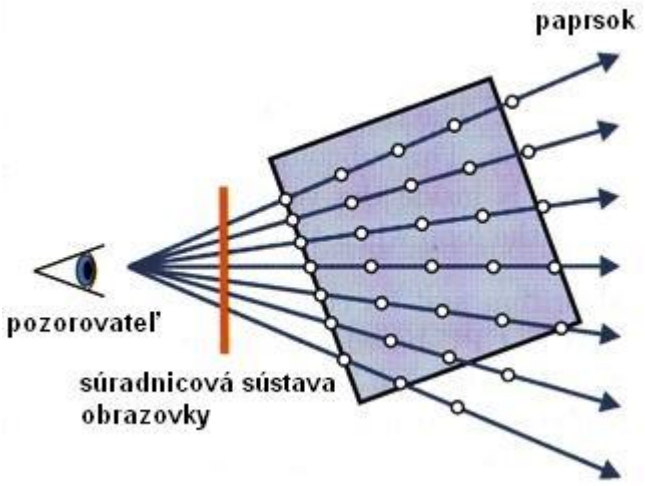
Najsofistikovanejším algoritmom zo všetkých doposiaľ spomenutých je tzv. *gray – level gradient shading*. Ten predpokladá, že na povrchu dochádza k najväčšej zmene hodnôt vzorkou a preto zisťuje normálu k povrchu v smere gradientu, t.j. v smere najväčšej zmeny hodnôt intenzity v objeme. V dôsledku veľkého rozsahu intenzít vytvára táto metóda hladké povrchy bez artefaktov (bod h) na obrázku č. 27).

3.2.3. Algoritmus vrhania paprskov (Ray Casting)

V predošlej podkapitole bol podaný stručný prehľad niektorých existujúcich metód, ktoré pracujú na princípe vrhania paprskov do scény a následným vyhodnotením intenzity jasu, resp. farby v podobe RGB pozdĺž ich trajektórie. V tejto podkapitole bude podrobne vysvetlené ako vlastne táto skupina algoritmov funguje. Metóda Ray Casting je zjednodušením známeho algoritmu Ray Tracing.

Hlavnou myšlienkou algoritmu vrhania paprskov je snaha priamo vyčísliť objemový integrál (kapitola 2.1.3) pozdĺž paprskov vyslaných z pozície pozorovateľa, resp. kamery. Každým pixlom na obrazovke, t.j. v súradnicovej sústave obrazovky, je do scény, resp. objemu vyslaný paprsok. Na obrázku č. 28 je možné názorne vidieť tento postup. Následne dochádza ku prevzorkovaniu objemových dát v diskrétnych pozíciách pozdĺž vrhnutého paprsku. Skalárne hodnoty získané v týchto diskrétnych pozíciách sa pomocou transfer functions mapujú na optické vlastnosti (farba, nepriehľadnosť). Tie sa pozdĺž parpsku pri jeho prechode objemom neustále akumulujú v procese kompozície. Kompozícia môže byť realizovaná v rovnakom smere ako je postup paprsku. Ide teda o kompozičné schéma typu spredu – dozadu (angl. front – to – back). Jeho podoba je:

$$C_{dst} = C_{dst} + (1 - \alpha_{dst})C_{src}$$
$$\alpha_{dst} = \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src}$$



Obrázok č. 28: Princíp algoritmu vrhania paprskov

Funkčnosť tohto algoritmu môže byť opísaná nasledujúcim pseudokódom:

```
Urči vstupnú pozíciu do objemu so skalárnymi dátami;
Vypočítaj smer každého parpsku;
While (pozícia parsku je v objeme)
{
      Načítaj skalárnu hodnotu na pozícii vzorku;
      Namapuj skalárnu hodnotu na farbu a priehľadnosť;
      Kompozícia farby a priehľadnosti;
      Postúp v smere paprsku o určitú vzdialenosť;
}
```

V súlade s daným pseudokódom je možné rozdeliť činnosť tohto algoritmu do nasledujúcich krokov:

Inicializácia paprsku:

V závislosti na parametroch kamery v scéne a pozícii konkrétneho pixla na obrazovke je potrebné inicializovať práve spracovávaný paprsok. Táto komponenta ma za úlohu vypočítať vstupný bod do objemu s objemovými dátami. Ide vlastne o súradnice bodu, ktorý je prvým priesečníkom paprsku a hraničnej obálky (angl. bounding box)[7] objemových dát. V tejto fáze sa takisto určí smer šírenia paprsku.

[7] Ohraničujúca plocha urýchľujúca výpočet priesečníkov objektov

Cyklus zodpovedný za prechod objemovými dátami:
Ide o hlavnú komponentu. Je zodpovedná za prechod objemovými dátami v smere paprsku, pričom dochádza k vyčísleniu objemového integrálu v diskrétnych pozíciách pozdĺž paprsku. Každá iterácia tohto cyklu pozostáva z nasledujúcich sub – komponent:

- **načítanie skalárnych dát** – dochádza k načítaniu skalárnej hodnoty z objemových dát v konkrétnej diskrétnej pozícii na dráhe paprsku, čo vedie k potrebe rekonštrukcie, resp. interpolácie skalárnych objemových dát (kapitola 2.3.2); z takto získanej skalárnej hodnoty je možné získať korešpondujúcu hodnotu farby a nepriehľadnosti pomocou klasifikácie (napr. s využitím tzv. look - up textúr)
- **kompozícia –** doposiaľ naakumulovaná hodnota farby a nepriehľadnosti je aktualizovaná vzhľadom k použitému kompozičnému schématu
- **posun objemom v smere paprsku** – z aktuálnej pozície na dráhe paprsku je potrebné prejsť na nasledujúci bod, resp. na ďalšiu diskrétnu pozíciu
- **ukončenie cyklu** – subkomponenta zodpovedná za ukončenie celého cyklu; overuje sa, či sa paprsok stále nachádza v objeme, resp. hraničnej obálke daného objemu; v prípade, že paprsok je stále v objeme, je možné vstúpiť do ďalšej iterácie cyklu

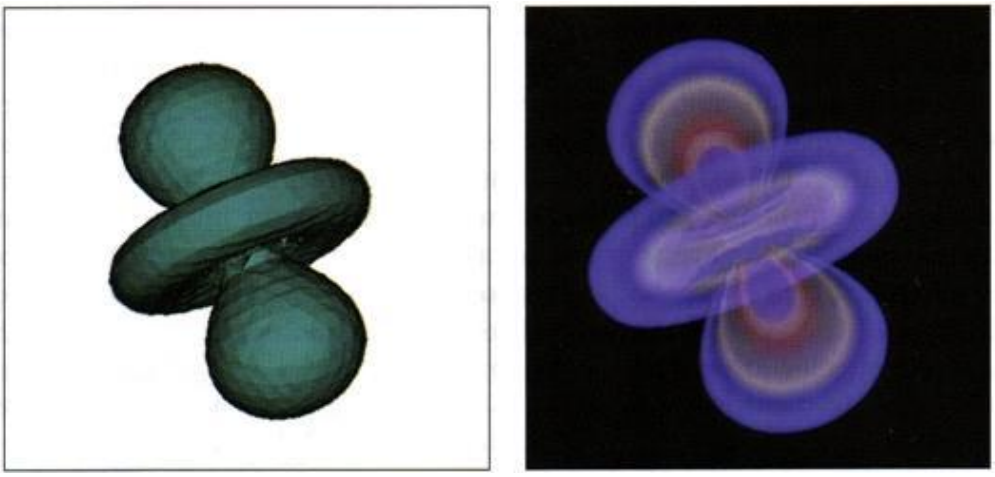
Práve popisovaný algoritmus je možné jednoducho a efektívne prispôsobiť ku implementácii na moderných GPU. Tie totiž umožňujú paralelné spracovanie, ktoré je vhodné práve pre činnosť tohto algoritmu. Prechod každého paprsku objemom je realizovaný jednou z viacerých hardwarových pipeline na GPU – nezávisle na iných paprskoch vyslaných do scény.

Existuje viacero techník, ako urýchliť beh tejto metódy. Jednou z nich je technika umožňujúca prerušiť transport paprsku objemom v prípade, že objemové elementy nachádzajúce sa ďalej v objeme od aktuálnej pozície sú zatienené, resp. zakryté. Tento stav nastane, keď sa hodnota nahromadenej veličiny α_{dst} blíži k 1. Ďalšou možnosťou ako urýchliť beh tohto algoritmu je dynamické prispôsobovanie veľkosti kroku pri prechode paprsku v závislosti na regiónoch v objeme, t.j. prázdne časti objemu je možné preskočiť.

Existujú modifikácie Ray Casting – u umožňujúce vykresľovat objemové dáta na iných než pravidelných mriežkach, Ide napr. o mriežky, v ktorých má voxel tvar štvorstenu.

Na nasledujúcich obrázkoch sa nachádzajú ukážky tohto algoritmu.

Obrázok č. 29: Ray Casting tých istých dát. Rozdiel spočíva v odlišnom počte krokov algoritmu a v aplikácii iného typu klasifikácie

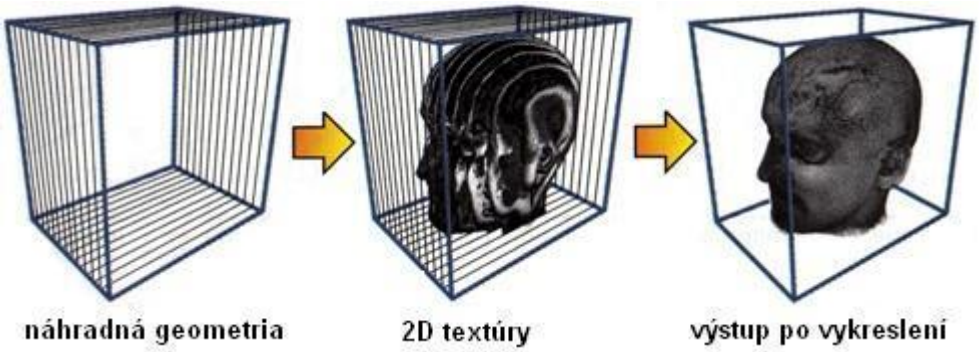


Obrázok č. 30: Ray Casting na mriežkách s voxelmi typu štvorsten

3.2.4. Texture Slicing

Jedná sa o najčastejšie nasadzovaný algoritmus zo skupiny algoritmov fungujúcich v súradnicovej sústave objektu. Táto metóda využíva fakt, že diskrétne trojrozmerné objemové dáta môžu byť reprezentované v podobe vysokého počtu navzájom rovnobežných dvojrozmerných plátov. Ide o akési nahradenie objemových dát tzv. náhradnou (angl. proxy) geometrickou reprezentáciou (viď. obrázok č. 31). V takomto prípade je možné vizualizovať objemové dáta pomocou vykresľovania vysokého počtu polopriehľadných dvojrozmerných plátov extrahovaných zo samotných skalárnych objemových dát. Pláty sa premietajú do zobrazovacej roviny a v závislosti na použitej kompozičnej schéme je možné objemové dáta vykresliť. Samotné objemové skalárne dáta sú uložené buď v plátoch, ktoré sú reprezentované dvojrozmernými textúrami alebo sú uchovávané v rámci trojrozmerných textúr[8].

[8] V takom prípade ide o trojrozmernú variantu texture slicing - u

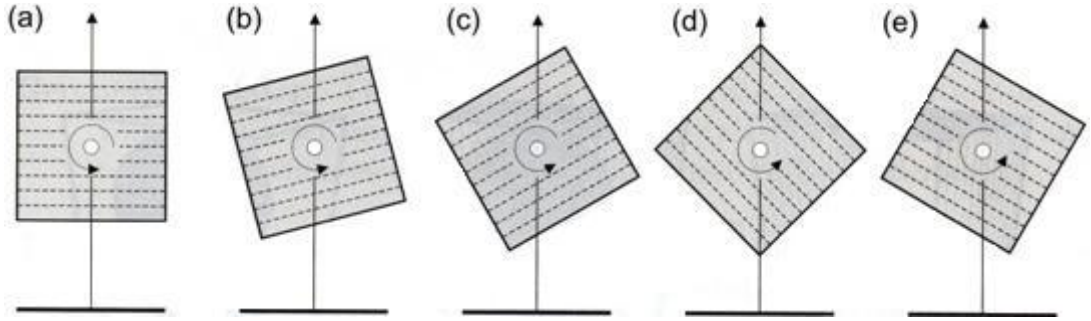


Obrázok č. 31: Reprezentácia objemu vysokým počtom rovnobežných plátov

2D texture slicing:

Prvý variant tohto algoritmu pracuje len s dvojrozmernými textúrami ako náhradnou geometriou, pričom využíva mohutne bilineárnu interpoláciu ako metódu rekonštrukcie skalárnych objemových dát. Objemové dáta sú teda uložené v podobe určitého počtu dvojrozmerných textúr (viď. predchádzajúci obrázok). Z toho vyplýva aj obmedzenie tejto metódy – táto technika je schopná zachytiť len dvojrozmernú podmnožinu v skutočnosti trojrozmerných objemových dát. Samotné textúry sú uložené ako zásobník plátov zarovnaných podľa všetkých troch súradnicových os x, y, z. Dôvod k tejto trojitej reprezentácii dát je ten, že pre prístup k textúram sa používajú len dvojrozmerné textúrové súradnice. Prvé dve slúžia na určenie pozície v rámci textúry, tretí koordinát je konštantný a určuje pozíciu textúry, resp. plátu v rámci konkrétneho vybraného zásobníka.

Užívateľ požaduje, aby bolo možné dané objemové dáta ľubovoľným spôsobom v scéne rotovať. Z toho dôvodu musí aplikácia neustále vyhodnocovať orientáciu, resp. smer plátov vzhľadom k pozícii pozorovateľa, resp. kamery. V takejto situácii je potrebné vybrať jednu z troch súradnicových os a to tým spôsobom, že sa minimalizuje uhol medzi normálou plátu a zamýšľaným smerom pozorovania v scéne. Takáto realizácia však vedie k významnému problému – paprsok môže prejsť pomedzi dva pláty bez toho, aby zasiahol čo len jeden z nich. Práve preto je potrebné disponovať až tromi zásobníkmi plátov, resp. dvojrozmerných textúr, každý zásobník pre jednu z hlavných súradnicových os v scéne. Počas behu aplikácie preto dochádza k prepínaniu medzi týmito zásobníkmi, čo vedie k nežiadaným artefaktom. Obrázok č. 32 názorne popisuje tento jav v dvojrozmernej variante. Objem je postupne rotovaný. V situácii, kedy ma uhol medzi normálou plochy a smerom pozorovania veľkosť 45 stupňov, nastáva nejednoznačný stav. Orientácia pre pláty je nejednoznačná a môže byť zvolená náhodne. Pri uhle väčšom než 45 stupňov musí dôjsť k prepnutiu, resp. výmene zásobníkov.



Obrázok č. 32: Prepínanie medzi trojicou zásobníkov s plátmi

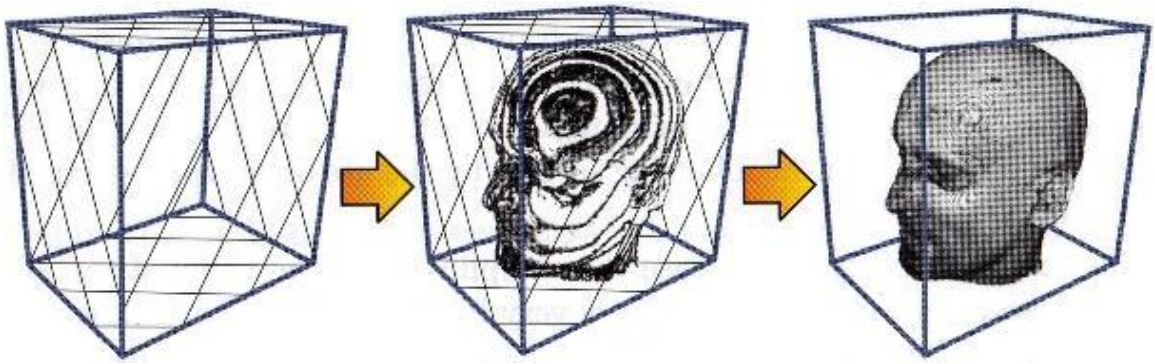
Táto metóda je extrémna rýchla, ale trpí výraznými nedostatkami. Jedným z nich je fakt, že počet plátov v zásobníku je fixný. Tým pádom nie je možné zvýšiť vzorkovaciu frekvenciu, čo môže viesť k vzniku artefaktov v obraze. Takisto nastávajú problémy, kedy dochádza k prepínaniu, resp. výmene jednotlivých zásobníkov. Na nasledujúcom obrázku je vidieť vznik aliasingu z dôvodu nedostatočného počtu plátov.


Obrázok č. 33: Aliasing spôsobený nedostatočnou vzorkovacou frekvenciou

3D texture slicing:

Viacero problémov predošlej metódy bolo spôsobených fixným počtom plátov a ich statickým zarovnaním v rámci súradnicovej sústavy. Navyše reprezentácia plátov pomocou dvojrozmerných textúr neumožňuje trilineárnu interpoláciu, ktorá je žiadanou možnosťou ako zvýšiť kvalitu výstupu. Použitie trojrozmerných textúr vedie k tomu, že už nie je potrebná dekompozícia objemových dát na rovinné pláty. Tým sa zvyšuje flexibilita akou je možné túto dekompozíciu realizovať.

Zásadným problémom predošlej metódy bola nekonzistentná vzorkovacia frekvencia, ktorá bola spôsobená statickým charakterom náhradnej geometrie. Pretože trojrozmerné textúry umožňujú orientovať pláty ľubovoľne v rámci trojrozmerného priestoru, je možné prispôsobovať vzorkovaciu frekvenciu, resp. vzdialenosť medzi plátmi dynamicky, vzhľadom k aktuálnemu smeru pozorovania objemových dát a to pre ľubovoľný smer, z akého scénu užívateľ sleduje. Takisto odpadá problém vzniku artefaktov spôsobených prepínaním medzi zásobníkmi s plátmi. Použijú sa totiž tzv. pláty zarovnané vzhľadom k smeru pozorovania (viď. obrázok č. 34). To znamená, že objemové dáta sú v každom okamihu vykresľovania narezané na pláty paralelne vzhľadom k rovine premietania. Z toho vyplýva, že náhradná geometria, resp. pláty, musia byť znova vypočítané v každom okamihu, kedy dôjde k zmene smeru pozorovania. S tým však nevyvstáva žiaden väčší problém pre programátora. Ten sa môže spoľahnúť na možnosti trojrozmerných textúr a grafickej karty, ktorá podporuje trilineárnu interpoláciu, čím dôjde k samotnému výpočtu plátov paralelných k rovine premietania.



Obrázok č. 34: Dekompozícia objemu na pláty zarovnané vzhľadom k smeru pozorovania

Táto metóda je určite sofistikovanejšia než jej predchodca. Odstraňuje takmer všetky nedostatky 2D texture slicing – u pričom ponecháva väčšinu z jej výhod. Oproti 2D texture slicing, 3D texture slicing nevyžaduje 3 kópie plátov zarovnaných podľa súradnicových osí. Podporuje dynamické prispôsobovanie vzorkovacej frekvencie, t.j. je možné meniť vzdialenosti medzi plátmi pomocou trilineárnej interpolácie.

4. Implementácia vlastnej aplikácie

V predošlých kapitolách tejto práce boli vysvetlené všetky teoretické pojmy a princípy, na ktorých stojí objemové vykresľovanie. Takisto boli predstavené najpoužívanejšie objemové algoritmy v praxi a tento text sa zaujímal mimo iné aj o všetky fázy, z ktorých proces objemového vykresľovania pozostáva.

V tejto kapitole budú nadobudnuté poznatky pretavené do podoby návrhu a implementácie aplikácie vykresľujúcej objemové dáta. Cieľom tejto časti textu bude podať informácie o všetkých aspektoch, ktoré sa vyskytli pri samotnom návrhu, resp. implementácii, či už ide o povahu zobrazovaných dát, spôsob realizácie alebo použité technológie.

4.1. Použité technológie

Aplikácia bola vyvíjaná v programovacom jazyku C++ vo vývojovom prostredí Microsoft Visual Studio 2008 Professional Edition s použitím grafickej knižnice OpenGL vo verzii 2.1. Pre návrh užívateľského rozhrania aplikácie bol použitý framework Qt v. 4.5.2.

4.2. Spôsob realizácie, použité prístupy

Činnosť programu je možné rozdeliť do niekoľkých krokov, resp. fáz. Samotná implementácia aplikácie vychádza z algoritmov Ray Casting a Texture Slicing. Ide vlastne o ich kombináciu, pričom sa využili výhody oboch techník. Nasledujúce podkapitoly textu sa venujú jednotlivým krokom, z ktorých navrhnutá aplikácia pozostáva.

4.2.1. Voxelizácia

Voxelizácia je prvá etapa navrhnutého algoritmu. V tejto fáze dochádza k prevodu spojitej reprezentácie telies na reprezentáciu objemovú. Tento krok je nutný z dôvodu, že samotné dáta, ktoré sa majú vizualizovať, sú reprezentované pomocou tradičnej, t.j. povrchovej reprezentácie (vo forme trojuholníkových sietí). Dáta v tejto podobe nemožno priamočiaro vykresľovať pomocou akýchkoľvek objemových techník, a to platí i pre vyvinutý algoritmus. Bolo teda potrebné tento prevod z jednej reprezentácie objektov do druhej naprogramovať.

Prvým krokom tohto prevodu je rozsekanie načítanej geometrie v scéne na rovnobežné pláty (analógia z 2D texture slicing). Zdrojový kód z nasledujúcej ukážky realizuje požadované rezanie:

```
    for (u16 planeIndex = 0; planeIndex < nSlices; planeIndex++)
    {
        for (u32 meshPartIndex = 0; meshPartIndex < meshParts.size(); meshPartIndex++)
        {
                if(!meshParts[meshPartIndex]->UnSerialize()) continue;
                if (planeIndex == 0) meshParts[meshPartIndex]->BeginCutting();

                meshParts[meshPartIndex]->Slice_Calc(planes[planeIndex]);
                meshParts[meshPartIndex]->Slice_GetResult(points, results);

                RayCastingEngine::getInstance()->preDraw(points, results, planeIndex);

                if (planeIndex == nSlices - 1) meshParts[meshPartIndex]->EndCutting();
        }
}
```

Výpis č. 1: Rezanie objektov v scéne na pláty

Samotné rezanie realizuje metóda `Slice_Calc`. Jej vstupným argumentom je pole rovín, určujúce v akom smere (na základe normály reznej roviny) a s akou hustotou rezanie prebehne. Reže sa vždy v smere kolmom na niektorú z hlavných súradnicových osí. To, ktorá z nich to bude závisí od veľkosti rezaného objektu pozdĺž týchto osí. Reže sa totiž vždy v smere, v ktorom je objekt najrozľahlejší. Výstupom voxelizácie, v tomto prípade rezania objektov na pláty, je pole `points` novovzniknutých vrcholov a pole `results` obsahujúce pre každý z vrcholov určitú skalárnu hodnotu v ňom uloženú. Je samozrejmé, že pri rezaní objektov dochádza k interpolácii všetkých per – vertex hodnôt, t.j. hodnôt asociovaných s daným vrcholom. Obe polia sú vstupom pre ďalší krok aplikácie.

4.2.2. Vykresľovanie plátov do 3D textúry

Jedná sa o najkomplexnejšiu a najzložitejšiu časť aplikácie. Dôkazom toho je aj to, že je možné túto fázu rozdeliť na ďalšie subkomponenty. Ako už bolo spomenuté, vstupom pre túto časť programu je geometria, ktorá vznikla po orezaní scény reznými rovinami. Ide konkrétne o dve polia obsahujúce jednotlivé body a k nim priradené skalárne dáta, reprezentujúce nejakú zobrazovanú veličinu.

Vykreslenie geometrie do float textúry:

Jednotlivé body sa postupne v poradí jeden plát za druhým vykresľujú do dvojrozmernej float textúry. Aby bolo možné dosiahnuť takéhoto postupu, je nutné presmerovať cieľ renderovania z klasického framebuffera[9] spravovaného operačným systémom, do užívateľsky definovaného framebuffera. Pre túto úlohu sa využila technika renderovania do tzv. Frame Buffer Object – u (ďalej len FBO). K FBO je pripojená užívateľsky definovaná 2D float textúra. V určitej dobe behu programu je FBO aktívny, čím dochádza k presmerovaniu vykresľovania do textúry k nemu pripojenej. Samotné vykresľovanie

[9] Ide o oblasť v pamäti veľkosti obrazovky reprezentovaná v podobe pixlov, do ktorej sa vykresľuje výstup

do textúry je určované pomocou vertex a fragment shaderov. Nasledujúci výpis predstavuje zdrojový kód oboch shaderov:

```
attribute float result;
varying float value;

void main(void)
{
value = result;
gl_Position = ftransform();
}
```

Výpis č. 2: Vertex shader zodpovedný za vykresľovanie do 2D float textúry

```
varying float value;

void main(void)
{
   gl_FragColor = vec4(value, 1.0, 0.0, 1.0);
}
```

Výpis č. 3: Fragment shader zodpovedný za vykresľovanie do 2D float textúry

Z výpisu oboch shaderov je jasné, že pri vykresľovaní vrcholov orezanej geometrie do 2D float textúry dochádza zo strany GPU k automatickej interpolácii skalárnych hodnôt uložených v týchto vrcholoch. To je jeden z hlavných dôvodov, prečo sa vôbec vykresľovanie do textúry použilo. Je samozrejmé, že pre korektné ukladanie interpolovaných skalárnych hodnôt je potrebné nastaviť v OpenGL rovnobežné premietanie. Pozícia kamery je takisto dôležitá. Rovnobežné premietanie má v tomto prípade prednosť pred perspektívou. Je totiž nežiaduce, aby dochádzalo k deformácii obrazu na základe pozície, resp. vzdialenosti plátu od pozorovateľa.

Je samozrejmé, že pre vykresľovanie takéhoto vysokého počtu bodov a per – vertex skalárnych hodnôt sa oplatí využiť tzv. vrcholové polia (angl. vertex arrays). Ide o vnútornú štruktúru OpenGL uchovávajúcu celú geometriu (vertexy a iné dáta) zabezpečujúcu efektívne a čo možno najrýchlejšie vykresľovanie.

Načítanie dát z float textúr do 3D objemovej textúry:

Všetky dáta uložené v predchádzajúcom kroku do dvojrozmerných float textúr sa v tejto etape načítajú do trojrozmernej float textúry. V tejto fáze programu táto textúra už reprezentuje, resp. uchováva samotné objemové dáta, ktoré je možné vykresliť do finálnej podoby na obrazovku.

4.2.3. Prechod objemovými dátami a ich finálne zobrazenie

Ide o posledný krok celej aplikácie vedúci k požadovanému výstupu. V predchádzajúcich fázach sa algoritmus približoval svojim správaním k Texture Slicing algoritmu. Teraz sa však situácia mení a vykresľovanie v tomto kroku je typickou ukážkou algoritmu Ray Casting.

Hlavnou náplňou je definitívne vykreslenie objemových dát reprezentovaných 3D float textúrou z predchádzajúceho kroku. Tento proces je realizovaný pomocou shaderov z nasledujúceho výpisu:

```
uniform vec3 scaleFactor;

void main(void)
{
    vec4 vertexPos = gl_Vertex;
    vec4 texturePos = gl_MultiTexCoord0;

    vertexPos *= vec4(scaleFactor, 1.0);
    texturePos *= vec4(scaleFactor, 1.0);

    gl_TexCoord[0] = texturePos;
    gl_Position = gl_ModelViewProjectionMatrix * vertexPos;
}
```

Výpis č. 4: Vertex shader zobrazujúci objemové dáta z 3D textúry

```
uniform sampler3D data;
uniform sampler1D transfFunc;
uniform float stepSize;
uniform int iterations;
uniform vec3 volExtMin;
uniform vec3 volExtMax;

void main(void) {

    float value;
    vec4 src = vec4(0.0);
    vec4 dst = vec4(0.0);

    vec3 position = gl_TexCoord[0].xyz;
    vec3 cam = gl_ModelViewMatrixInverse[3];
    vec3 direction = gl_TexCoord[0].xyz - cam;
    direction = normalize(direction);

    for (int i = 0; i < iterations; i++) {

            value = texture3D(data, position).x;

            src = texture1D(transfFunc, value);
```

```
            src.w *= 0.75;
            src.xyz *= src.w;

            dst = (1.0 - dst.w) * src + dst;

            position = position + direction * stepSize;

            vec3 v1 = sign(position - volExtMin);
            vec3 v2 = sign(volExtMax - position);

            float isInside = dot(v1, v2);

            if (isInside < 3.0 || dst.w > 0.95) break;
        }

        gl_FragColor = dst;
}
```

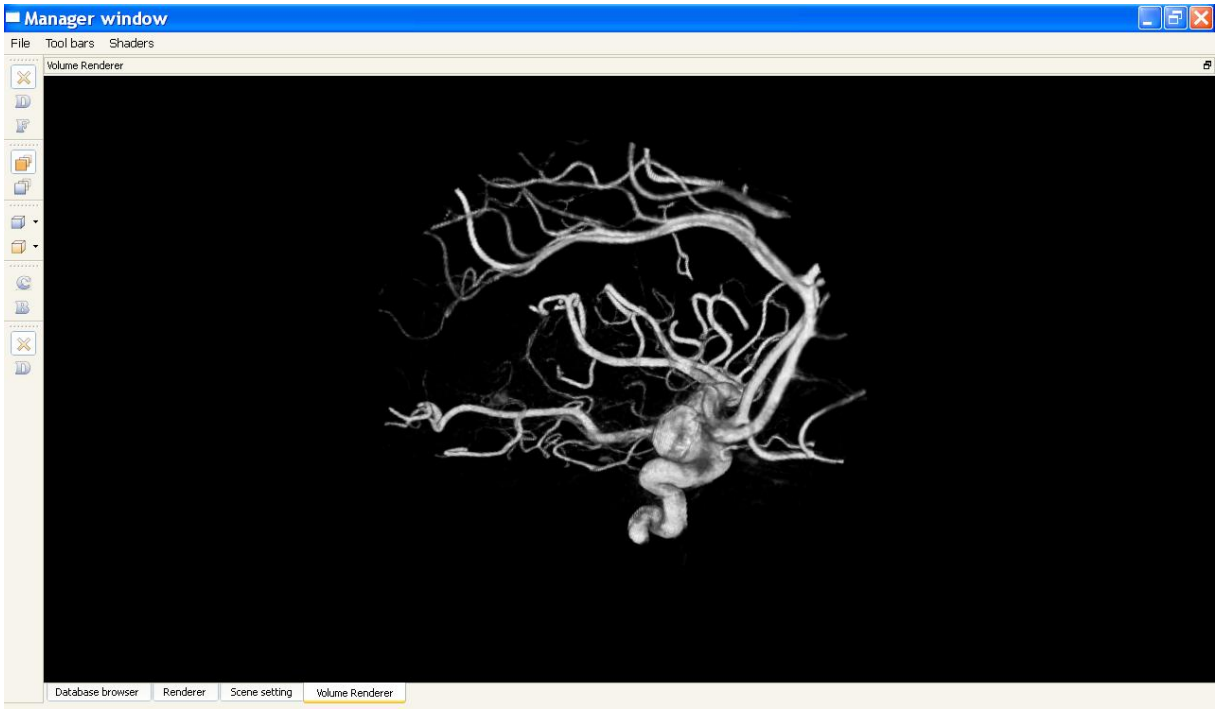
Výpis č. 5: Fragment shader zobrazujúci objemové dáta z 3D textúry

Na tomto mieste je vhodné zamerať sa v stručnosti na činnosť príslušného fragment shaderu. Zaujímavá je hlavne časť nachádzajúca sa v cykle `for`. Ide o hlavnú časť shaderu. Je zodpovedná za prechod objemovými dátami v smere paprsku, pričom dochádza k vyčísleniu objemového integrálu v diskrétnych pozíciách pozdĺž paprsku Takto získané skalárne hodnoty sa v procese klasifikácie pomocou gradientnej textúry, v tomto prípade textúry označenej ako `transfFunc`, mapujú na farbu a nepriehľadnosť.

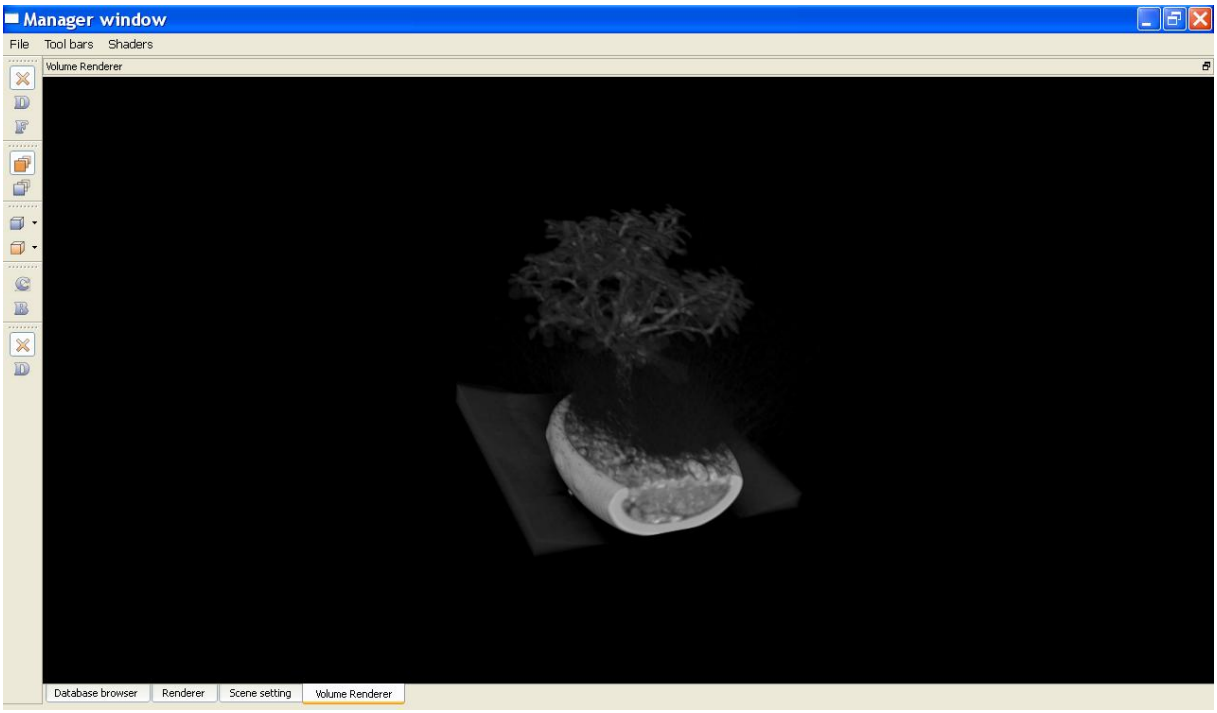
Klasifikácia prebieha nasledovne. V konkrétnom bode objemu sa vyčísli objemový integrál, čím je možné získať v tomto bode skalárnu hodnotu. Aplikácia má však pre jednotlivé skalárne hodnoty, ktoré sa vyskytujú v objemových dátach pripravené mapovanie, resp. prevod, ako takúto hodnotu previesť najčastejšie na farbu v podobe RGB. Tento prevod funguje tak, že jednotlivé skaláry v objeme a im prislúchajúca hodnota RGB definovaná programátorom tvoria body, cez ktoré sa preloží krivka, v tomto prípade kubický splajn. Tým sa docieli interpolácia medzi takýmito bodmi, resp. medzi ich RGBA hodnotami. Z týchto interpolovaných hodnôt sa zostrojí gradientná 1D textúra, ktorá sa použije pri mapovaní vo fragment shaderi.

4.3. Ukážky výstupov aplikácie

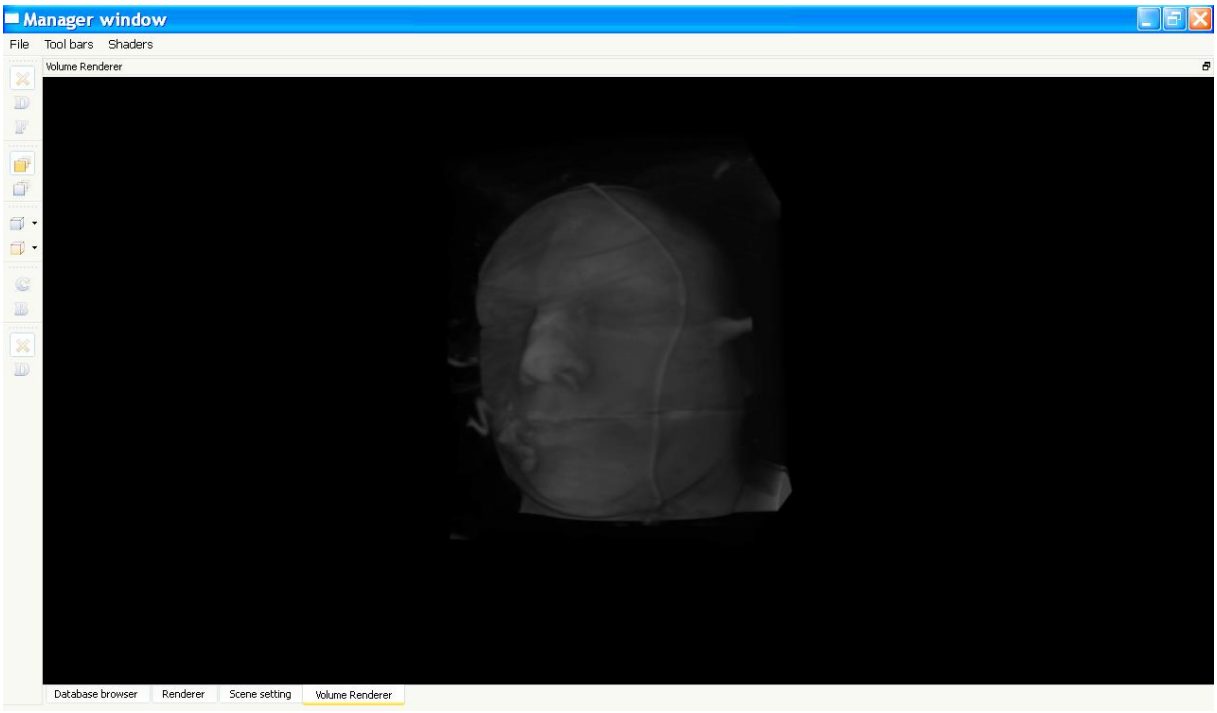
Táto kapitola prezentuje naprogramovanú aplikáciu prostredníctvom ukážkových výstupov (angl. screenshots). Tie je možné vidieť na nasledujúcich obrázkoch:



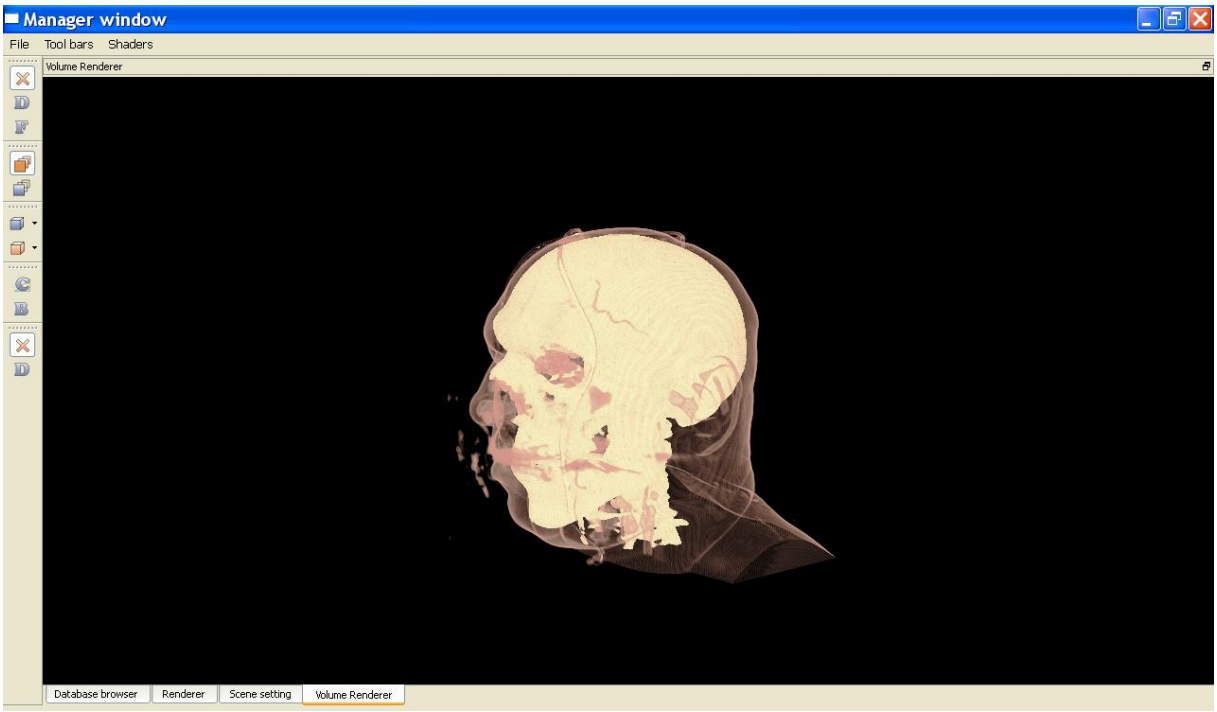
Obrázok č. 35: Aneurisma vykreslená naimplementovanou aplikáciou



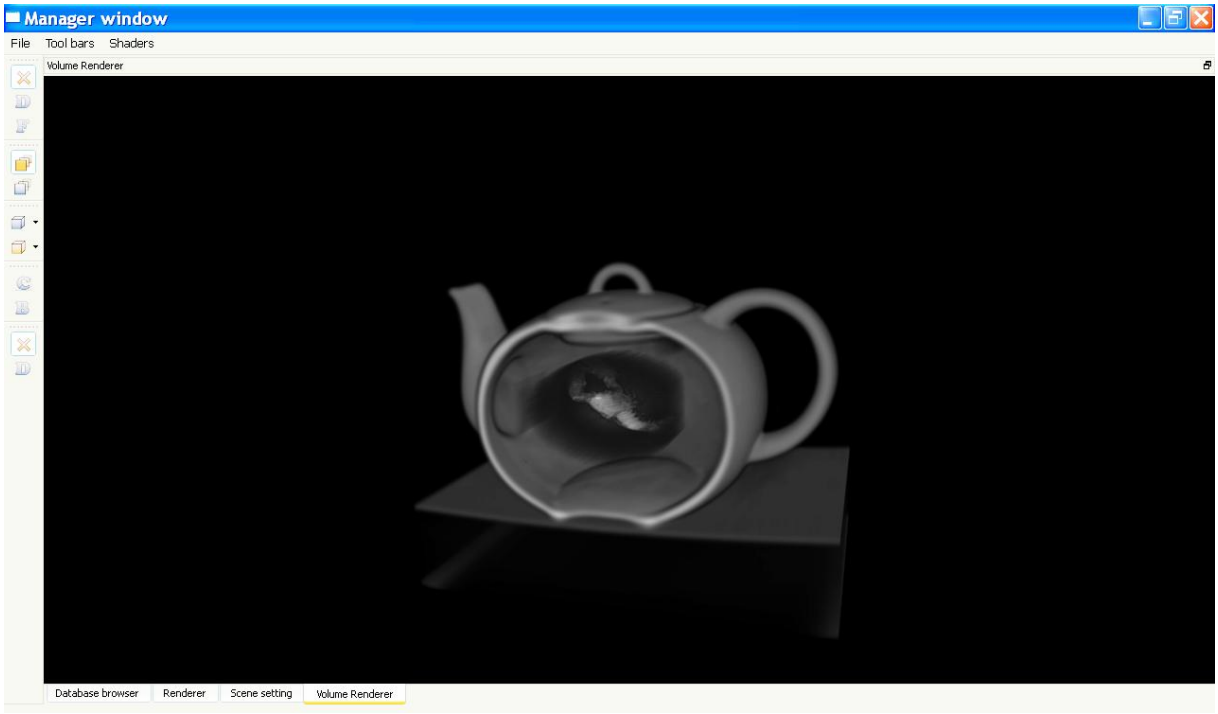
Obrázok č. 36: Strom Bonsai vykreslený naimplementovanou aplikáciou



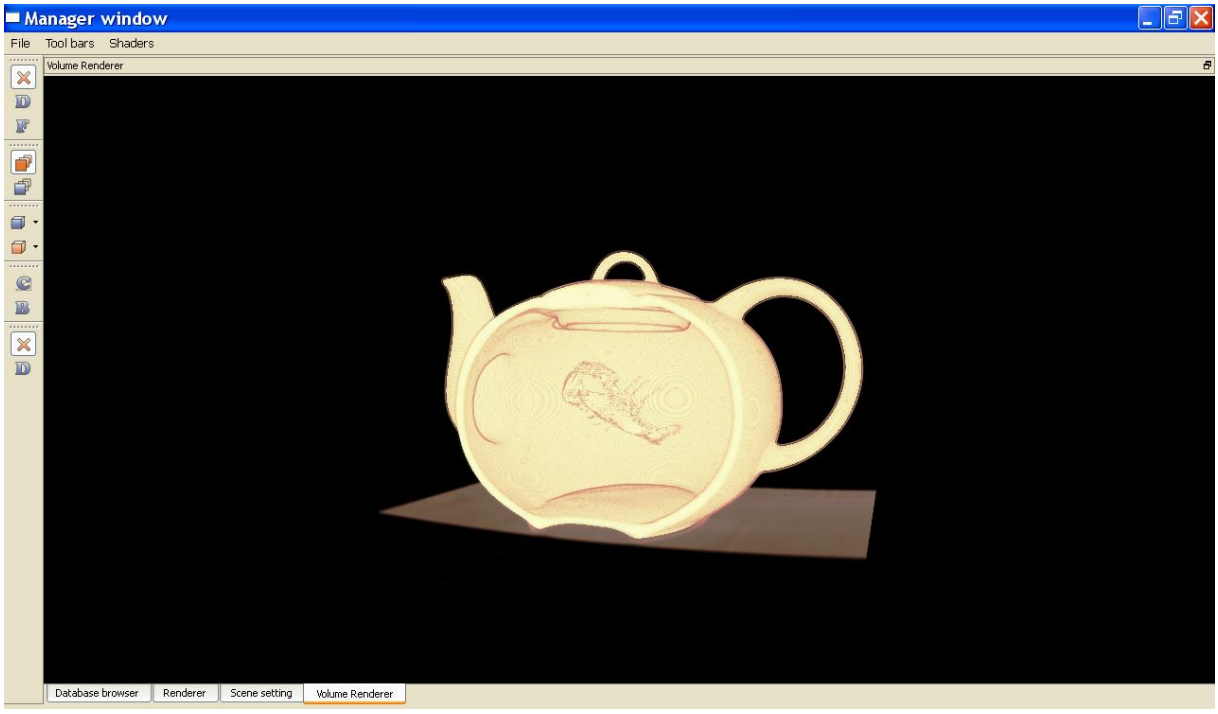
Obrázok č. 37: Ľudská hlava vykreslená v šedotónovom prevedení



Obrázok č. 38: Tie isté dáta ako na predošlom obrázku ale v RGBA prevedení



Obrázok č. 39: Čajník so morským rakom uprostred



Obrázok č. 40: Tie isté dáta ako na predošlom obrázku, ale v RGBA prevedení

5. Záver

Hlavnými cieľmi tejto diplomovej práce bolo zmapovanie možností vykresľovania objemových dát a následný návrh a implementácia vlastnej aplikácie pre OpenGL. Myslím si, že vytýčené ciele diplomovej práce sa mi vo väčšej miere podarilo splniť. Podrobne som rozobral všetky aspekty súvisiace s objemovým vykresľovaním. Išlo najmä o rovnicu šírenia svetla v priestore a jej vzťah k objemovému integrálu. Spomenul som všetky najčastejšie používané aproximácie, resp. modely umožňujúce reálnym aplikáciám numericky vyčísliť objemový integrál a v reálnom čase vykresliť objemové dáta. Takisto som sa zameral na pôvod a charakteristiku samotných objemových dát. Uviedol som, na akých typoch diskrétnych mriežok sú reprezentované a aké problémy z danej reprezentácie vyplývajú. Podrobne som popísal problém dvojitej interpretácie voxelov a v neposlednom rade som sa zameral na problém rekonštrukcie, resp. interpolácie objemových dát v priestore. Kapitoly venujúce sa konkrétnym objemovým algoritmom poskytujú state – of – the – art v danej oblasti počítačovej grafiky. Podrobne som sa venoval spoločným znakom činnosti všetkých algoritmov v podobe volume – rendering – pipeline. V neposlednom rade som všetky nadobudnuté znalosti pretavil do podoby návrhu a implementácie vlastnej aplikácie. Na tomto mieste musím uviesť, že sa mi síce podarilo naprogramovať danú aplikáciu, ale jej migrácia do prostredia existujúcej aplikácie na vizualizáciu dát so zameraním na oblasť konečných prvkov, sa mi už nepodarila realizovať. Napriek tomu je vyvinutý program zástupcom v súčasnosti najpoužívanejšej techniky vykresľovania objemových dát – ide o metódu známu ako Ray Casting.

Pri písaní tejto diplomovej práce som si uvedomil, akým búrlivým rozvojom objemové modelovanie za niekoľko rokov prešlo. Aj v súčasnosti je možné sledovať neustály posun v tejto oblasti. Je preto možné v budúcnosti výsledky tejto práce rozšíriť o ďalšie prístupy. Vychádzajúc z implementácie aplikácie, vidím určité možnosti najmä v oblasti klasifikácie. Tento problém nie je v odborných kruhoch stále vyriešený na vyhovujúcej úrovni. Konkrétne sa jedná o problematiku návrhu užívateľsky prijateľných tzv. transfer functions.

Téma tejto diplomovej práce bolo pre mňa maximálne pútavé. Nadobudol som mnoho zaujímavých informácií a získal skúsenosti s vývojom aplikácii vykresľujúcich objemové dáta.

Citovaná literatúra

1. ENGEL, Klaus, et al. *Real Time Volume Graphics*. 1st edition. India : A K Peters, Ltd., 2006. Physical Model of Light Transfer, s. 4-8. ISBN 1-56881-266-3.

2. ENGEL, Klaus, et al. *Real Time Volume Graphics*. 1st edition. India : A K Peters, Ltd., 2006. Discretization, s. 10-14. ISBN 978-1-56881-266-3.

3. ENGEL, Klaus, et al. *Real Time Volume Graphics*. 1st edition. India : A K Peters, Ltd., 2006. Reconstruction, s. 21-24. ISBN 978-1-56881-266-3.

4. ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. druhé vydání. Brno : Computer Press, 2004. Radiance, s. 324-325. ISBN 80-251-0454-0.

5. ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. druhé vydání. Brno : Computer Press, 2004. Mřížky, s. 255-256. ISBN 80-251-0454-0.

6. ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. druhé vydání. Brno : Computer Press, 2004. Skalární objemové algoritmy, s. 459-461. ISBN 80-251-0454-0.

7. ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. druhé vydání. Brno : Computer Press, 2004. Přímé zobrazování objemů, s. 461-464. ISBN 80-251-0454-0.

8. HAYWARD, Kyle. *Graphicsrunner.blogspot.com* [online]. 2009 [cit. 2010-03-15]. Volume Rendering. Dostupné z WWW: <http://graphicsrunner.blogspot.com/2009/01/volume-rendering-101.html>.

9. TRIERS, Peter. *Www.daimi.au.dk/~trier* [online]. 2009 [cit. 2009-10-07]. GPU raycasting. Dostupné z WWW: <http://www.daimi.au.dk/~trier/?page_id=98>.

10. STEGMAIER, Simon; STRENGERT, Magnus; KLEIN, Thomas. *Wwwvis.informatik.uni-stuttgart.de* [online]. 2005 [cit. 2010-03-19]. A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting. Dostupné z WWW: <http://wwwvis.informatik.uni-stuttgart.de/eng/research/fields/current/spvolren/>.

11. SONG, Ho Ahn. *Www.songho.ca* [online]. 2005 [cit. 2010-02-01]. OpenGL Frame Buffer Object (FBO). Dostupné z WWW: <http://www.songho.ca/opengl/gl_fbo.html>.

Zoznam obrázkov

A. Obsah DVD

Obsahom DVD priloženého k diplomovej práci je:

- adresár so skompilovanou spustiteľnou aplikáciou
- moje zdrojové súbory
- text diplomovej práce vo formáte Word a PDF
- ukážky výstupov z aplikácie
- readme.pdf – dokument obsahujúci bližšie informácie o ovládaní aplikácie
- ďalšie potrebné súbory k spusteniu